Maxime GRESSIER
Master 2 Astrophysics & Datascience
School year 2024-2025

MASTER 2 INTERNSHIP REPORT

17 FEBRUARY - 18 JULY 2025

PINN-RT EoR: A Physically Informed Neural Network for Radiative Transfer in the Epoch of Reionization

---

Observatoire astronomique
de Strasbourg

Internship at
Observatoire Astronomique de Strasbourg
11 rue de l'Université
67000 Strasbourg, FRANCE

Supervisor : Pierre Ocvirk (pierre.ocvirk@astro.unistra.fr)
Co-Supervisor : Emmanuel Franck (emmanuel.franck@inria.fr)

Aknowledgements

I would like to thank Pierre and Emmanuel for their great supervision and all their help during this internship.
I also thank Mei and Joubine for their kind support during the project.
A special thanks to my mother, Nathalie, for always being there and supporting me. I also want to thank my friends from the Master's program in Strasbourg — Camille, Samuel, Lucille, Nicolas, and Baptiste — for their help and encouragement during this time.

Abstract

In cosmological simulations, radiative transfer is often performed using the M1 method because of its ease of implementation and computational efficiency. However, it results in a family of artifacts linked to its fluid-like treatment of photons. More accurate approaches, such as the Pn method, offer better physical fidelity but are significantly more demanding in terms of computational time and memory footprint. In this work, we propose a neural-network-based alternative. We investigate a neural networks approach that leverages radiative transfer equations to predict the system's evolution without relying on training data. This report introduces the method, referred to as the discrete Physically Informed Neural Network (discrete PINN), and tests it on simple scenarios where the M1 method fails to produce realistic results. We demonstrate that our method can yield qualitatively satisfactory outcomes opening the way to more extended future implementations and studies.

Contents

1 Introduction

Shortly after the Big Bang, the Universe consisted of a hot plasma of nucleons and electrons, with temperatures too high to allow electrons to bind with hydrogen nuclei. As the Universe expanded, the energy of photons decreased until the temperature dropped to approximately $10^4 K$, allowing electrons to combine with protons to form the first atoms, predominantly hydrogen. This also led to a significant drop in the density of free electrons, reducing the interactions between photons and electrons. Consequently, photons were able to travel freely through the plasma. This event, known as recombination, began at a redshift of $z \approx 1100$ and marked the point when the Universe became mostly neutral.

At recombination, the Universe became transparent to cosmic microwave background (CMB) photons but remained opaque to the UV ionizing photons. The epoch that followed is referred to as the Dark Ages, during which the gas was neutral and transparent, but no luminous sources had yet formed. The Universe continued to expand passively until the onset of the reionization epoch.

Around a redshift of $z \approx 20$, overdensities in the neutral gas led to the formation of the first stars. Notably, the James Webb Space Telescope (JWST) has detected galaxies at redshift as high as 16 (Atek et al. 2022; Helton et al. 2025; Adams et al. 2022). An illustration of this epoch of reionization is given in Figure 2. The radiation emitted by these stars began to ionize the surrounding medium, creating bubbles of ionized gas that eventually percolated and overlapped, leading to the complete reionization of the intergalactic medium by $z \approx 6$ (Barkana & Loeb 2001; Aghanim et al. 2020; Bosman et al. 2022). This period also corresponds to the formation of the first galaxies and quasars.

Several methods are available to probe the epoch of reionization. One of the most prominent is the Gunn-Peterson trough, which involves studying the spectra of distant quasars (Becker et al. 2001). These objects are extremely luminous, making them observable at very large distances, even during the epoch of reionization. Moreover, their intrinsic spectra consists of the spectral features of intervening neutral hydrogen clouds. Neutral hydrogen efficiently absorbs Lyman-alpha photons, and due to cosmic redshift, this absorption produces a characteristic trough in the observed spectra. This structure, known as the Lyman-alpha forest, can also be observed in the spectra of other bright sources, such as Gamma-Ray Burst (GRB) afterglows (Lamb & Haiman 2003).

However, analyzing these spectra is challenging due to significant noise resulting from the extreme distances of the sources. Furthermore, such observations are rare—especially for GRBs, which fade within just a few days. Therefore, developing efficient and accurate simulations is essential for advancing our understanding of the formation of the first stars and galaxies.

In astrophysics, it is essential to confront theoretical models with observational data. However, in some cases, such comparisons are not possible due to the lack of observations, often caused by the limitations of current technology in detecting extremely distant objects. This is particularly true for the study of the epoch of reionization, which remains extremely challenging to observe.

Furthermore, as previously mentioned, indirect analyses based on features such as the Gunn-Peterson trough or the Lyman-alpha forest are difficult to interpret due to their complexity and the low signal-to-noise ratio. As a result, simulations play a crucial role not only in advancing our theoretical understanding, but also in helping us interpret the complex data, such as the spectra of objects originating from the reionization epoch.

Another promising probe of the epoch of reionization is the 21 cm line of neutral hydrogen, as described in Shimabukuro et al. (2022). This spectral line corresponds to the hyperfine transition between the parallel and antiparallel states of the hydrogen atom in its fundamental level. Although this transition is rare, it provides a valuable tool to study this epoch. The SKA shown in Figure 1.

Figure 1: `https://www.skao.int/en/explore/telescopes`: SKA telescope in South Africa

In this context, simulations play a crucial role, with radiative transfer lying at the heart of modeling the epoch of reionization. Indeed, simulating this epoch involves modeling the formation of the first stars and the subsequent ionization of the surrounding neutral intergalactic medium. Therefore, the key physical process to solve is the radiative transfer equation, which can be written as follows:

$$\partial_t I(t, \mathbf{x}, \mathbf{v}) + \mathbf{v} \cdot \nabla_x I = -\kappa(t, \mathbf{x}, \mathbf{v}) I + S(t, \mathbf{x}, \mathbf{v}), \tag{1}$$

where $I(t, \mathbf{x}, \mathbf{v})$ represents the specific intensity of radiation, κ is the absorption coefficient, and S denotes the source function.

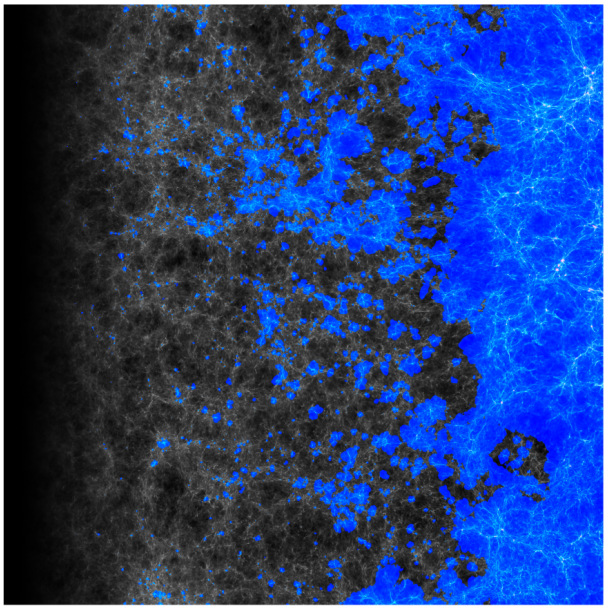


Figure 2: Ocvirk et al. (2020): Illustration of the epoch of reionization resulting of the CoDa II simulation.

Several methods already exists to solve this equation, among which the M1 method (Aubert & Teyssier 2008) is the most widely used in cosmological simulations with codes such as RAMSES or DYABLO. For instance, Ocvirk et al. (2020) employed RAMSES in the CoDa II simulation to model reionization and galaxy formation, the result of this simulation is shown Figure 2. The M1 method involves computing the first two moments of the radiative transfer equation, yielding the conservation equations for radiation energy and flux.

$$\partial_t E + \nabla_x \mathbf{F} = -\kappa c E + S, \tag{2}$$

$$\partial_t \mathbf{F} + c^2 \nabla_x \mathbf{P} = -\kappa c \mathbf{F}. \tag{3}$$

To fully solve the system of equations and compute the radiation energy density E and flux $\mathbf{F}$, the M1 method requires a closure relation, originally proposed by Levermore (1984), which provides the Eddington tensor necessary to determine the radiation pressure tensor $\mathbf{P}$. This closure is straightforward to implement and computationally inexpensive, making it particularly suitable for cosmological simulations. However, it relies on crude approximations that can lead to inaccuracies. For example, when two photon beams cross, the M1 closure averages them, resulting in an incorrect photon flux direction.

This limitation is illustrated in the left panel of Figure 3, taken from the paper Palanque et al. (2025) which is still in prep, where the two crossing beams are smoothed into a single diagonal beam, in contrast to the distinct crossing beams shown in the right panel. Another artifact induced by the M1 closure is the appearance of pseudo-sources between two isotropic sources. This effect can be seen in Figure 4, where the overdensity between the two isotropic sources in the left panel represents a spurious source absent in the right panel.

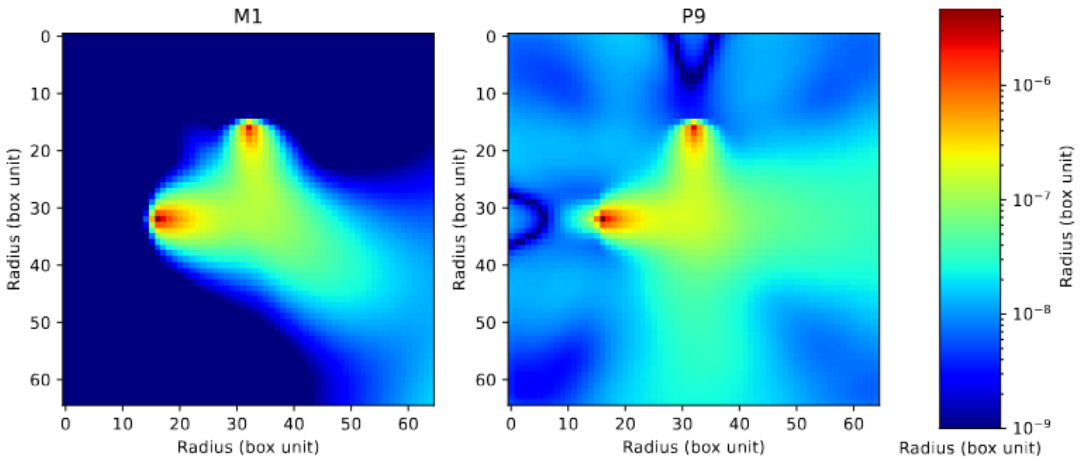


Figure 3: Palanque et al. (2025): Simulation of two crossing beams with M1 (left panel) and P9 (right panel)

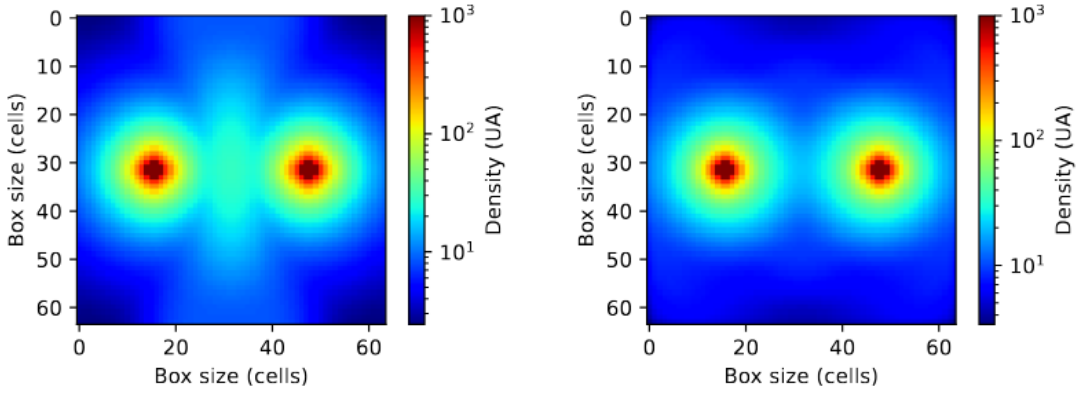


Figure 4: Palanque et al. (2025): Simulation of two isotropic sources with M1 (left panel) and P9 (right panel)

Alternative methods, such as the Pn method, can avoid the artifacts introduced by the M1 closure. This approach solves a higher number of moments of the radiative transfer equation (1), allowing for a more accurate description of the angular distribution of the radiation field. As shown in the right panels of Figures 3 and 4, the Pn method produces results that match the expected physical behavior, correctly capturing the crossing beams and avoiding the formation of pseudo-sources. However, the main drawback of the Pn method is its high computational cost which makes it difficult to apply in large-scale cosmological simulations.

To address the issue of computational cost, neural networks offer a promising alternative. As demonstrated in Franck et al. (2025), neural networks are capable of performing simulations within a realistic computational time, even in high-dimensional settings—something that is often challenging for traditional numerical methods (see also for applications in different physical contexts Raissi et al. (2019); Beltran-Pulido et al. (2022); Zhang et al. (2023)). This makes it particularly interesting to explore the use of neural networks for solving the radiative transfer equation, aiming to avoid the artifacts introduced by the low number of moments computed by M1 while maintaining computational efficiency.

The objective of this internship is to simulate the radiative transfer equation without the absorption term, focusing on the transport component of the equation:

$$\partial_t I(t, \mathbf{x}, \mathbf{v}) + \mathbf{v} \cdot \nabla_x I = S(t, \mathbf{x}, \mathbf{v}). \tag{4}$$

We will reproduce the two benchmark tests—beam crossing and isotropic sources—using discrete Physically Informed Neural Networks (PINNs). The goal is to evaluate the method's applicability, accuracy, performance, and memory footprint.

We begin by introducing the concept of neural networks through the simple case of the Multilayer Perceptron, which is the architecture used in this work. Section 3 is dedicated to a detailed description of the methodology employed to simulate the radiative transfer equation using neural networks. This includes a presentation of the projection method in Section 3.1, as well as an introduction to Physically Informed Neural Networks. The section concludes with a description of the computational framework and hardware used in this study, presented in Section 3.6.

The results are presented in Section 4, where we show the evolution of test systems using the PINN approach, as well as the outcomes of the benchmark tests designed to highlight the known limitations of the M1 method. Finally, we discuss the performance of the PINN method relative to M1 in Section 5.

2 Basics on Neural Networks

There are many types of neural networks designed for various applications, such as object classification from images (e.g., galaxy classification), image generation, or large language models (LLMs) for text generation. In our case, we aim to use a neural network for solving partial differential equations (PDEs). To this end, we employ a simple and widely used architecture: the Multilayer Perceptron (MLP).

As illustrated in Figure 5, an MLP consists of at least three layers: an input layer, an output layer, and one or more hidden layers. Hidden layers are composed of units called neurons. The number of hidden layers and the number of neurons per layer are hyperparameters that can be chosen by the user. Each element (input, neuron, or output) is fully connected to all elements in the adjacent layers. These connections are associated with weights w, and each neuron in a hidden layer also has an associated bias b.

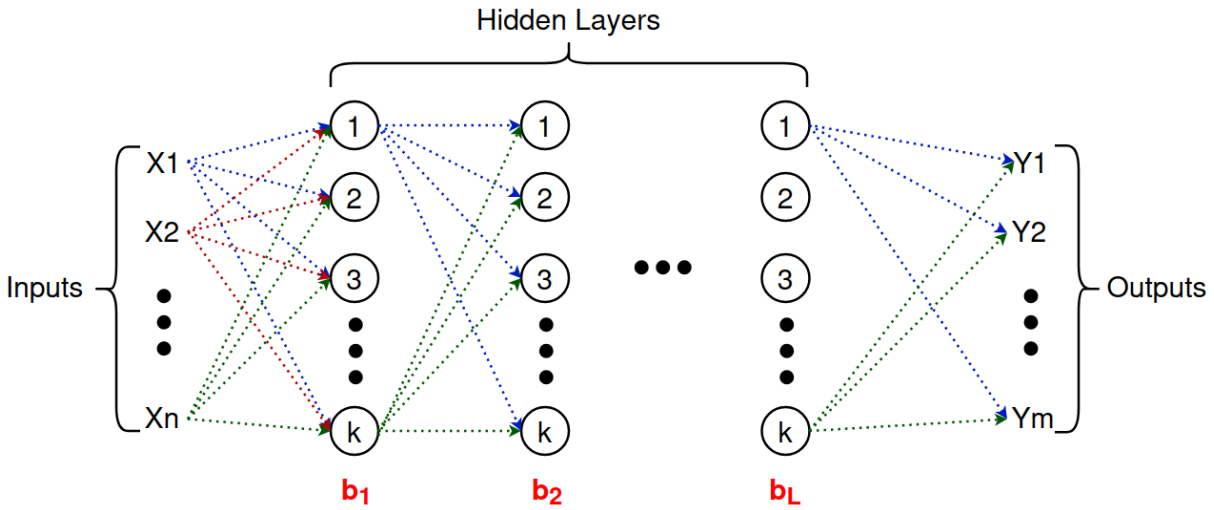


Figure 5: Diagram of a multilayer perceptron

To compute the value of each neuron in the next layer (or in the output layer), the following formula is applied:

$$y_i = \sigma\left(\sum_{j=0}^{N} w_{ij}x_j + b_i\right), \tag{5}$$

where σ is the activation function, a non-linear function that determines whether a neuron is activated. For example a widely used activation function in neural networks method is the *ReLu* function (Rectified Linear Unit) given by:

$$ReLu(x) = \max(0, x), \quad \forall x \in \mathbb{R}.$$

Other classical function used in machine learning are the sigmoid function given by:

$$sigmoid(x) = \frac{1}{1 + e^{-x}}, \quad \forall x \in \mathbb{R}.$$

The use of non-linearity through the activation function enables neural networks to approximate and solve non-linear problems, which is essential for many physical applications. As we will discuss in Section 3.3, activation functions can also be used to enforce positivity on the network outputs.

By propagating information from the inputs to the outputs using this mechanism, the neural network produces a result aligned with the intended task-for instance, class probabilities in a classification problem, or, in our case, values corresponding to physical quantities at given coordinates.

Classically, to obtain the correct set of parameters that produce accurate outputs, the network must undergo a training phase. This training involves comparing the predicted outputs to a set of reference data. The discrepancy between the prediction and the target is quantified using a loss function, denoted $L(\theta)$, where θ represents the current set of parameters (weights and biases). PINNs, on the other hand, do not require a training dataset. Instead, they are guided by the PDEs of the problem. It compare directly the approximated solution with a solution of those PDEs and adjust the parameters to minimize the differences.

We determine iteratively the value of the parameters θ by optimizing the loss function. This can be done using classical gradient descent methods, but the optimization can have a strong impact on the solution. The specific implementation of this minimization strategy in the context of our problem will be detailed in Section 3.2.1.

In Section 3.2, we will see how discrete PINNs circumvent the need for training datasets by using the PDE of the system at hand.

3 Methodology

As previously mentioned, the objective of this internship is to approximate the solution of the radiative transfer equation (1) using a neural network.

A neural network consists of neurons arranged in one or more layers, where each neuron is connected to the next layer through weighted links. Each connection carries a weight, and each layer includes a bias. The entire set of weights and biases is denoted by θ, which defines the neural network. The network output, in our case the predicted intensity $I_\theta(t, x, v)$, depends on this set of parameters.

Our goal is to find the optimal set of parameters θ such that: $I_\theta(t, x, v) \approx I(t, \mathbf{x}, \mathbf{v})$

In this section, we will explain how to optimize this set of parameters when we don't know the solution $I(t, \mathbf{x}, \mathbf{v})$.

3.1 Projection Method

The task of approximating the solution $I_\theta(t, \mathbf{x}, \mathbf{v})$ with a neural network $I(t, \mathbf{x}, \mathbf{v})$ can be interpreted as a projection problem. Indeed, our goal is to minimize the discrepancy between the neural network's output and the true solution, which amounts to projecting the function represented by the neural network onto the solution space.

To find the optimal set of parameters θ that minimizes this difference, we seek to solve the following optimization problem:

$$\theta = \underset{\theta}{\operatorname{argmin}} \int_{\Omega, S^1} \left| I_\theta(t, \mathbf{x}, \mathbf{v}) - I(t, \mathbf{x}, \mathbf{v}) \right|^2 d\mathbf{x} d\mathbf{v}. \tag{6}$$

This expression corresponds to the minimization of the L^2 norm of the error over the spatial domain Ω and the angular domain S^1, which is a standard approach in projection-based approximation methods.

In practice, we choose to evaluate and minimize the residual only at a finite number n of points, referred to as collocation points. The minimization problem can thus be reformulated as:

$$\theta = \underset{\theta}{\operatorname{argmin}} \sum_{j=1}^{n} \left| I_\theta(t, \mathbf{x}_j, \mathbf{v}_j) - I(t, \mathbf{x}_j, \mathbf{v}_j) \right|^2, \tag{7}$$

where the $(\mathbf{x}_j, \mathbf{v}_j)$ are the collocation points. They can be chosen randomly within the computational domain (as in our case) or on a grid.

3.2 Physically Informed Neural Networks

In our case, we do not have access to the solution $I(t, \mathbf{x}, \mathbf{v})$ (Allaire et al. 2018), which prevents us from directly comparing our neural network approximation with an exact solution as described in Section 3.1. However, we do know the PDE governing the system, which in our case is Equation (4). Moreover, we set the initial state of the system by imposing $I(t = 0, \mathbf{x}, \mathbf{v}) = I_0(\mathbf{x}, \mathbf{v})$ and we will optimize the parameters of the set corresponding to the first time step to reproduce this condition in the initialization step.

The system can thus be formulated as:

$$\begin{cases} \partial_t I + \mathbf{v} \cdot \nabla_x I = f(t, \mathbf{x}, \mathbf{v}) \\ I(t = 0, \mathbf{x}, \mathbf{v}) = I_0(\mathbf{x}, \mathbf{v}). \end{cases} \tag{8}$$

This formulation allows us to adopt a physics-informed approach, where the neural network is optimized not by fitting to data, but by enforcing the satisfaction of the PDE and the initial condition.

In this section, we explain how to formulate this problem using a neural network and how to optimize its parameters to approximate the desired solution.

We have several solutions to this problem. There are several strategies to tackle this task. One option is to treat time as a continuous variable, just like the spatial coordinates, and to directly approximate a global function $I_\theta(t, \mathbf{x}, \mathbf{v})$ that satisfies the partial differential equation throughout the entire spatio-temporal domain. This approach is known as a continuous-time PINN.

Another possibility is to discretize the time domain and optimize the neural network iteratively at each time step. In this case, the network predicts the solution at time t_{i+1} based on the output from time t_i. This approach, which we adopt in this work, is referred to as a discrete PINN.

3.2.1 Continuous-Time PINNs

As previously mentioned, in the continuous-time PINN approach, the time domain is treated in the same way as the spatial and velocity domains. The neural network is optimized to reproduce the solution over the entire time domain within a single optimization process. To achieve this, we define a loss function that consists of two components:

1. A term that enforces the initial condition by measuring the discrepancy between the network's output at $t = 0$, denoted $I_\theta(0, \mathbf{x}, \mathbf{v})$, and the prescribed initial condition $I_0(\mathbf{x}, \mathbf{v})$.
2. A term that enforces the PDE by comparing the left-hand side of the equation, evaluated using automatic differentiation on the neural network, to the right-hand side $f(t, \mathbf{x}, \mathbf{v})$.

The optimal set of network parameters θ is obtained by minimizing this loss function. By definition, the condition for optimality is given by:

$$\nabla_\theta L(\theta) = 0. \tag{9}$$

The optimized set of parameters is the θ such that:

$$\begin{aligned}\theta = \underset{\theta}{\operatorname{argmin}} \Bigg(& \sum_{j=1}^{n} \left| \partial_t I_\theta(t, \mathbf{x}_j, \mathbf{v}_j) + \mathbf{v}_j \cdot \nabla_x I_\theta(t, \mathbf{x}_j, \mathbf{v}_j) - f(t, \mathbf{x}_j, \mathbf{v}_j) \right|^2 \\ & + \lambda_0 \sum_{j=1}^{n} \left| I_\theta(0, \mathbf{x}_j, \mathbf{v}_j) - I_0(\mathbf{x}_j, \mathbf{v}_j) \right|^2 \Bigg) = \underset{\theta}{\operatorname{argmin}}\, L(\theta),\end{aligned} \tag{10}$$

where λ_0 is the learning rate we assign to the optimization of the initial condition.

During the minimization, we randomly initialize the weights and biases, which together form the first set of parameters θ_0. At each minimization iteration, the loss function is evaluated for the current set of parameters, and the next set is obtained via a gradient descent step:

$$\theta_{k+1} = \theta_k - \eta A^{-1} \nabla_\theta L(\theta), \tag{11}$$

where η is the learning rate — a tunable hyperparameter that controls how much the network updates its parameters in response to the computed loss. This value must be large enough to ensure fast convergence, but small enough to avoid overshooting the minimum.

The matrix A^{-1} represents a preconditioning of the parameter space and encodes information about the geometry of the loss landscape. In our case, A corresponds to the Fisher metric, and the descent is performed using the natural gradient method. For more details about this approach and the structure of matrix A, we refer the reader to Nurbekyan et al. (2023).

According to Chen et al. (2023), this approach may lead to issues related to time causality. Indeed, by optimize over all time steps simultaneously — including the initial condition — the neural network might not sufficiently focus on the early stages of the evolution. As a result, it may fail to accurately capture the initial dynamics of the system, which are crucial for the correct propagation of the solution over time.

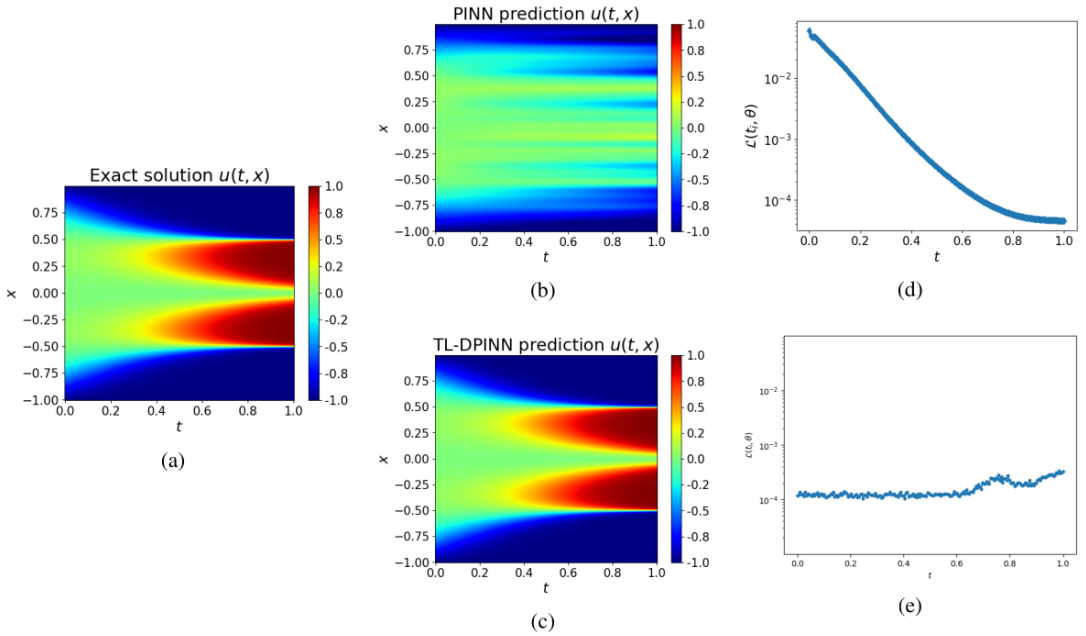


Figure 6: Chen et al. (2023): Comparison of the continuous-time PINN and discrete PINN methods for the simulation of the Allen-Cahn equation.

We can see in Figure 6 the solution of the Allen–Cahn equation, a reaction–diffusion equation that describes phase separation processes in multi-component alloy systems. It is clear that the solution approximated by the continuous-time PINN method is less accurate than the one obtained with the discrete PINN method. Moreover, as shown in the evolution of the loss functions in the right-hand plots, the continuous-time PINN appears to be less effectively optimized during the first time steps, which negatively affects the subsequent time steps as well.

3.2.2 Discrete PINNs

A possible solution to address this issue is to discretize the time domain and optimize a separate neural network for each time step. In other words, we aim to find a set of parameters θ_t for each time t, so that we can evaluate each time step with a specific neural network optimized for this specific time step, ensuring a faithful reproduction of the temporal evolution. This approach is known as the discrete PINN method, and it is the one we adopt for our tests.

To facilitate the optimization at each time step, we do not reinitialize the weights and biases randomly. Instead, for a given time step t_{i+1}, we initialize the neural network with the parameters θ_{t_i} which have already been optimized to approximate the solution at time ti t_i. We then optimize the new set of parameters $\theta_{t_{i+1}}$ by minimizing the following loss function:

$$\theta_{t_{i+1}} = \underset{\theta}{\operatorname{argmin}} \left(\sum_{j=1}^{n} \left| \Big(I_\theta(\mathbf{x}_j, \mathbf{v}_j) - I_{\theta_{t_i}}(\mathbf{x}_j, \mathbf{v}_j) + \Delta t \mathbf{v}_j \cdot \nabla_x I_{\theta_{t_i}}(\mathbf{x}_j, \mathbf{v}_j) \Big) - \Delta t f(t_i, \mathbf{x}_j, \mathbf{v}_j) \right|^2 \right),$$

$$L_{i+1}(\theta) = \sum_{j=1}^{n} \left| \Big(I_\theta(\mathbf{x}_j, \mathbf{v}_j) - I_{\theta_{t_i}}(\mathbf{x}_j, \mathbf{v}_j) + \Delta t \mathbf{v}_j \cdot \nabla_x I_{\theta_{t_i}}(\mathbf{x}_j, \mathbf{v}_j) \Big) - \Delta t f(t_i, \mathbf{x}_j, \mathbf{v}_j) \right|^2 \quad (12)$$

which can be noted:

$$\theta_{t_{i+1}} = \underset{\theta}{\operatorname{argmin}}\, L_{i+1}(\theta). \quad (13)$$

This loss function measures the discrepancy between the evolution predicted by the neural network — computed using the PDE, the parameters from the previous time step, and the current parameters — and the expected evolution of the system, given by the source term $f(t_i, \mathbf{x_k}, \mathbf{v_k})$. Using the same gradient descent approach as for continuous PINNs, we can update the parameter values to minimize the loss function:

$$\theta_{t_{i+1}}^{k+1} = \theta_{t_{i+1}}^{k} - \eta A^{-1} \nabla_\theta L_{i+1}(\theta). \quad (14)$$

As the minimization of the parameters θ_t at each time step depends on the parameters obtained at the previous step, it is crucial to ensure that the initial condition is accurately approximated. A well-approximated initial condition guarantees that the subsequent optimization steps are based on a reliable approximation of the solution.

To achieve this, the initialization step consists in minimizing a dedicated loss function that measures the discrepancy between the imposed initial condition $I_0(\mathbf{x}, \mathbf{v})$ and the approximation provided by the neural network with parameters θ_0.

$$\theta_0 = \underset{\theta}{\operatorname{argmin}} \sum_{j=1}^{n} \left| I_\theta(\mathbf{x}_j, \mathbf{v}_j) - I_0(\mathbf{x}_j, \mathbf{v}_j) \right|^2 = \underset{\theta}{\operatorname{argmin}}\, L_0(\theta). \quad (15)$$

As before, the parameters are updated using a gradient descent method:

$$\theta_0^{k+1} = \theta_0^k - \eta A^{-1} \nabla_\theta L_0(\theta). \quad (16)$$

Algorithm 1 summarizes the basic principle of the discrete PINN. In our implementation, the stopping criterion for the minimization loop at each time step is based on a fixed number of iterations rather than on a convergence threshold for the error.

Algorithm 1 Discrete PINN's principle

Input: Initial source I_0 and target equation to solve (8), number of time step for the evolution N_t.
PINN's Parameters: Number of iterations for the initialization minimization N_0, number of iterations for the optimization of each time step N, learning rate η, number of layer and number of neurons per layer and the threshold error ϵ.
Initialization step:
for k = 1 **to** N_0 **do**
    Compute the loss function $L_0(\theta)$ (15);
    Update the parameters using gradient descent (16);
end for
Evolution step:
for i = 0 **to** $N_t - 1$ **do**
    for k = 1 **to** N **do**
        Compute the loss function $L_{i+1}(\theta)$;
        Update the parameters using gradient descent (14);
        if $|L_{i+1}(\theta) - L_{i+1}(\theta)| < \epsilon$ or $k \leq N$ **then**
            Break;
        end if
    end for
end for
Output: Set of optimized θ_t for $t \in [0, N_t]$ to represent the evolution at each time step t.

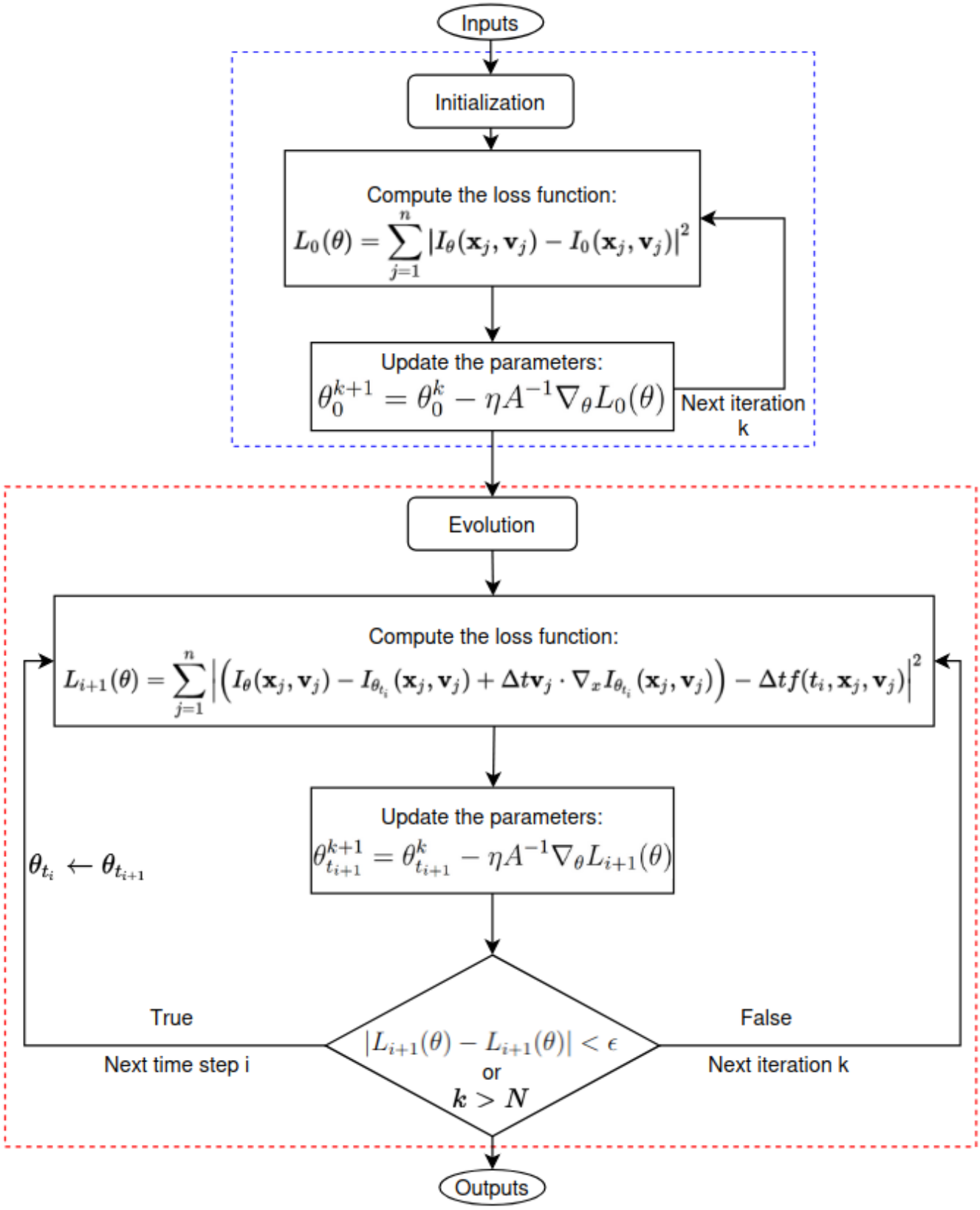


Figure 7: Schematic algorithm of the discrete PINN method.

3.3 Positivity

Naturally, neural networks do not inherently enforce constraints on the range of the output values. As a result, the predicted photon intensity can take negative values, which is not physically meaningful. To address this issue and ensure that the predicted intensity remains positive, we introduce suitable activation functions.

Activation functions in machine learning provide non-linearity and flexibility to neural networks. They are applied after each hidden layer to transform the weighted sum of inputs (including biases) into an output that allows the network to approximate complex non-linear functions. In our implementation, we use a sine activation function for the hidden layers, which was selected after testing several alternatives.

To enforce positivity of the final output, we apply a second activation function at the output layer. This function ensures that the predicted intensity values are strictly positive, leading to physically consistent results. The chosen output activation function is the softplus function, defined by:

$$softplus(x) = \log(1 + e^x). \tag{17}$$

This function behaves like the identity function for large x, while ensuring strictly positive outputs, thus providing a smooth alternative to the ReLU function.

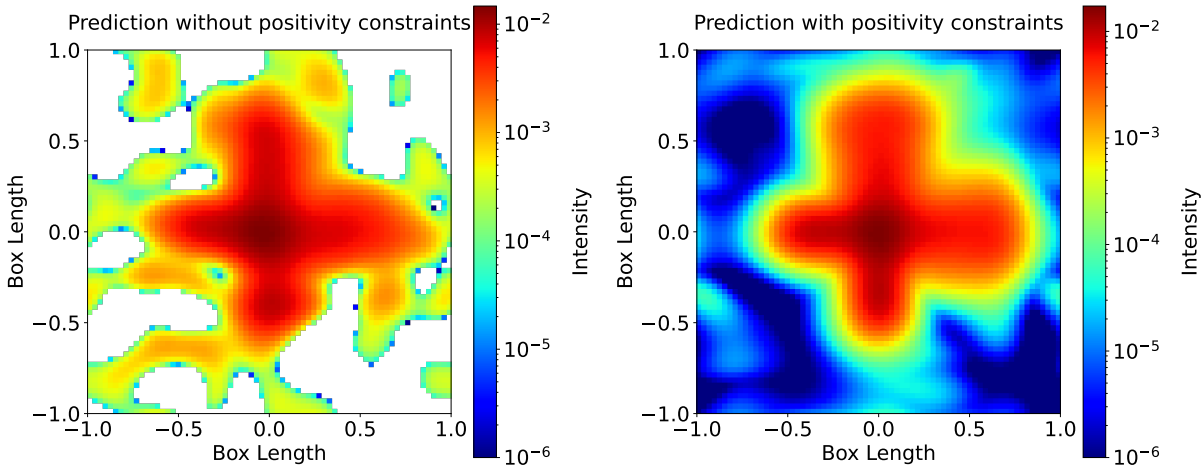


Figure 8: Simulation of crossing beams for the case where we impose the softplus activation function at the output to impose the positivity (right panel) and without this constraint (left panel).

In Figure 8, the left panel shows the result of the optimization when the neural network was not constrained to produce only positive values. Since we plotted the logarithm of the predicted intensity, regions where the network predicted negative values appear as white areas, as the logarithm is not defined for negative arguments. When we enforce positivity using the output activation function, the entire domain becomes visible, allowing us to interpret and evaluate the solution more reliably. We can then compare it with the semi-analytical solution described in Section 3.4, which is shown for the case of crossing beams in the top-right panel of Figure 13.

3.4 Analytical Solution

To compare the results obtained with the discrete PINN and to evaluate the accuracy of our method, we use a semi-analytical solution derived from the reference book Allaire et al. (2018). This solution is given by:

$$I(t,x,v) = I_0(x - tv, v) + \int_0^t S(s, x + (s-t)v, v)ds, \tag{18}$$

where $I_0(t,x,v)$ is the initial condition provided to the neural network, and $S(t,x,v)$ is the source term.

The integral term is computed using a piecewise constant approximation over time, which provides a simple and efficient way to evaluate the solution.

3.5 Periodic Boundary Conditions

As in cosmological simulations, tests are often performed with periodic boundary conditions, we need to implement such conditions in our neural network. To achieve this, we use a periodic embedding.

During optimization, at each iteration, the neural network evaluates the intensity using its current weights and biases, before updating these parameters according to the following equation:

$$I_{\theta^k}(t,\mathbf{x},\mathbf{v}) = \sigma\left(\sum_{l=1}^{N_{layer}}\sum_{m=1}^{N_{neurons}} w_{lm}^k X(t,\mathbf{x},\mathbf{v}) + b_l^k\right), \tag{19}$$

with X representing the inputs tensor, σ is the activation function, w_{lm}^k is the m^{th} neurons of the l^{th} layer at the k^{th} iteration and b_l^k is the bias of the l^{th} layer at the k^{th} iteration.

The periodic embedding consists in modifying the positions to ensure they remain within the spatial domain before computing the intensity at each step:

$$I_{\theta^k}(t,\mathbf{x},\mathbf{v}) = \sigma\left(\sum_{l=1}^{N_{layer}}\sum_{m=1}^{N_{neurons}} w_{lm}^k X(t,\cos(2\pi * \mathbf{x} + \phi_{lm}^k),\mathbf{v}) + b_l^k\right), \tag{20}$$

where the phase ϕ_{lm}^k is introduced as an additional parameter to be trained.

3.6 Framework and Mesocenter

In this section, we describe the software and hardware environments used to carry out this work.

3.6.1 SCIMBA Framework

Concerning the software used in this work, we relied on SCIMBA, a framework developed at the Advanced Mathematics Research Institute (IRMA) in Strasbourg by a team of researchers including Emmanuel Franck, Matthieu Boileau, and Victor Michel-Dansac. SCIMBA is an open-source project publicly available on GitLab at the following address: `https://gitlab.inria.fr/scimba/scimba`.

SCIMBA offers a collection of tools and examples dedicated to machine learning and neural networks methods for solving PDEs. The framework is built using the PyTorch library and provides implementations of various approaches such as continuous-time and sequential-time solvers, including PINNs, discrete PINNs, and the Neural Galerkin method, among others.

In this work, GIT was used to create a new branch of the code and provide modifications to adapt the existing neural network class and discrete PINN method to our case where we consider the phasespace and to implement the radiative transfer equation to test the discrete PINN in our context.

3.6.2 High Performance Computing Environment

Regarding the hardware, all computations were performed on the High Performance Computing (HPC) infrastructure of the University of Strasbourg. In particular, we made use of the GPUs available within the HPC cluster to accelerate the optimization of our neural networks. The cluster provides access to a variety of GPU models, including NVIDIA GTX 1080 Ti (11.1 GB), Quadro RTX 5000 (16.1 GB), Quadro RTX 6000 (22.7 GB), as well as several Tesla-class GPUs designed for large-scale computations.

To access these shared resources, we used the Slurm workload manager, a queue-based system that manages job submissions among multiple users. Each job must be submitted via the `sbatch` command, which requires specifying the computational resources needed (such as the number of nodes, GPUs, memory, and wall-time) in a dedicated job script. A typical submission command would be:

```
sbatch launch_RT_croise.txt
```

Here, `launch_RT_croise.txt` is a Slurm batch script that defines the execution parameters, including the script or Python code to run, the number of GPUs requested, the expected computation time, and the relevant resource partition. The script of the file `launch_RT_croise.txt` can be found in the appendix A.

4 Results

4.1 Crossing Gaussian Pulses

In this first test, we aim to analyze how the neural network reacts to a pulse in time. To this end, we define the initial source as two Gaussian distributions: one located on the left side of the domain and the other at the bottom. The left Gaussian is associated with a velocity distribution whose mean is oriented to the right, while the bottom Gaussian has a mean velocity directed upward.

The neural network used for this test consists of three hidden layers of forty neurons each. The activation function is a sine function, and to ensure positivity of the predicted intensity, we apply a softplus activation function at the output, as described in Section 3.3.

The optimization procedure is divided into two phases:

- An initialization step of 500 iterations using 30,000 collocation points randomly sampled within the domain.
- An evolution step composed of 100 iterations per time step, using 15,000 collocation points each.

The simulation is run up to a final time of $T = 0.95$, with a time step $\Delta t = 0.02$.

Figure 9 illustrates the evolution of the two pulses. The top row shows the prediction given by the discrete PINN at six regularly spaced time steps, while the bottom row displays the corresponding reference solution.

In this case, since the source is not continuous in time, we do not need to compute the integral term described in Section 3.4. The analytical solution is derived directly from the propagation of the initial condition. Because both position and velocity distributions are Gaussian, the solution at a given time t can be expressed as the product of a spatial Gaussian centered at $\mathbf{x} - \mathbf{v}t$ and a velocity Gaussian.

This allows for a simple computation of the analytical solution by updating the spatial Gaussian with time. During the evolution, we observe that the two pulses propagate in their respective directions, meet in the center of the domain, and then continue moving apart.

The formula for the left gaussian is as follows:

$$f_1(t, \mathbf{x}, \mathbf{v}) = e^{-(x_x - v_x t + 0.5)^2 \times 10} \times e^{-(x_y - v_y t)^2 \times 10} \times e^{-(v_x - 1)^2 \times 20} \times e^{-v_y^2 \times 20}, \tag{21}$$

and for the bottom gaussian we have:

$$f_2(t, \mathbf{x}, \mathbf{v}) = e^{-(x_x - v_x t)^2 \times 10} \times e^{-(x_y - v_y t + 0.5)^2 \times 10} \times e^{-v_x^2 \times 20} \times e^{-(v_y - 1)^2 \times 20}, \tag{22}$$

where in both functions, the factor 10 corresponds to the inverse of the squared dispersion in space and 20 is the inverse of the dispersion in velocity.

We expect to observe two photon packets propagating along their respective axes and crossing each other at the center of the box without interacting.

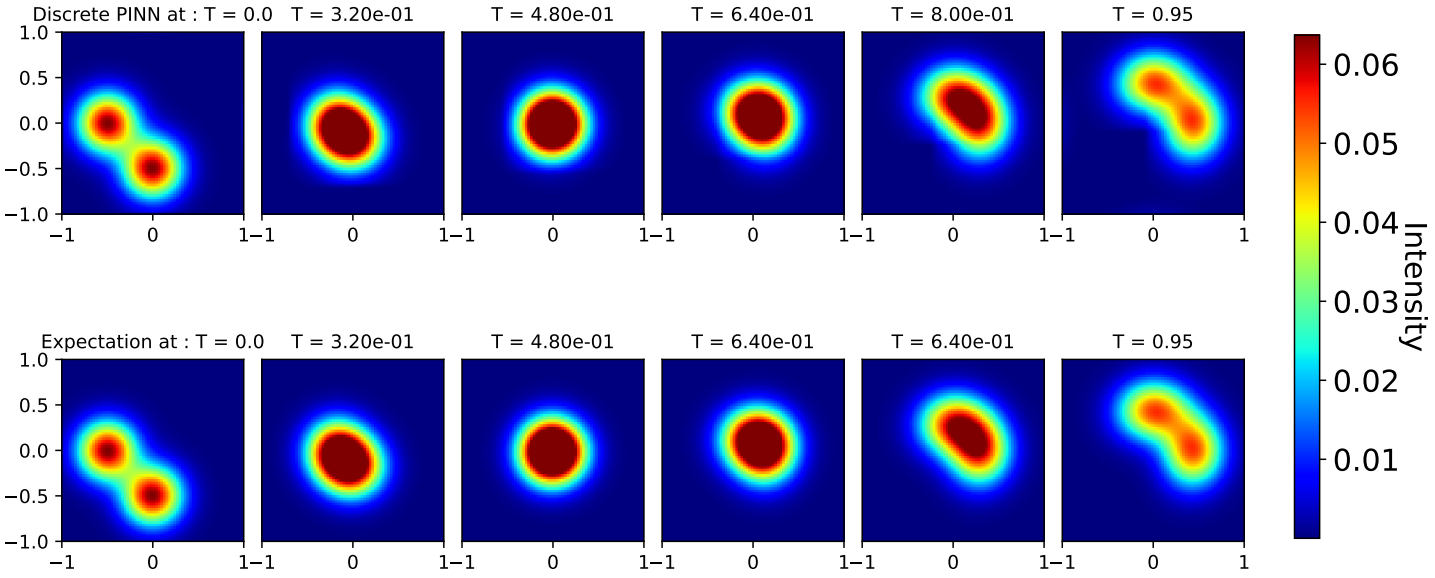


Figure 9: Evolution of two gaussian pulses approximated by the discrete PINN in first row and the solution is shown in the second row.

The discrete PINN appears to produce results in good agreement with the analytical solution. To verify this, we compute the relative L^2 norm of the error at each displayed time step. This evolution is shown in Figure 10. We observe that, throughout the evolution, the relative error remains on the order of magnitude of 1.5–2.5×10^{-2}.

There is, however, a noticeable gap between the error at the initial time (corresponding to the initialization step) and the error at the first time step of the evolution. This discontinuity can be explained by the difference in how these two steps are optimized.

Indeed, the initialization step is approximated using more iterations and a higher number of collocation points than a single evolution step. Moreover, the loss functions used in these two stages are different. For the initialization, the loss function simply measures the difference between the initial source term and the prediction of the neural network—it only needs to reproduce the initial condition I_0. On the other hand, the loss function used during the evolution step corresponds to the residual of the PDE, i.e., the difference between the PDE approximation given by the neural network and the expected evolution governed by the equation.

Because these two optimization steps are inherently different, a gap in accuracy may naturally appear. This highlights the importance of a well-converged initialization: since each time step depends on the previous one, it is crucial that the initial condition is strongly optimized to ensure accurate results throughout the entire simulation.

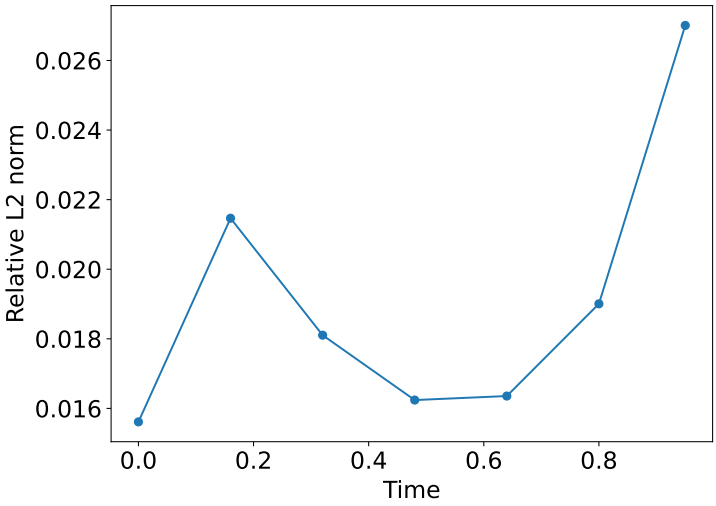


Figure 10: Evolution of the relative L2 norm at each time step

In Figures 11 and 12, we present the histograms of the differences between the predictions given by the neural network and the reference values from the exact solution, respectively after the initialization step and after one evolution step.

In Figure 11, corresponding to the initialization step, we observe a clear peak centered near zero. Most values lie within the range of approximately -2×10^{-3} to 3×10^{-3} and the peak is between -5×10^{-4} and 5×10^{-4}, indicating a relatively low error and a good fit to the initial condition.

In Figure 12, which corresponds to the evolution step, we also see a peak around zero. However, this peak is slightly more spread out, particularly towards negative values, with a tail extending between 0 and approximately -3×10^{-3}. This indicates that, although the predictions remain close to the exact solution, the evolution step introduces a slightly larger dispersion in the prediction errors compared to the initialization.

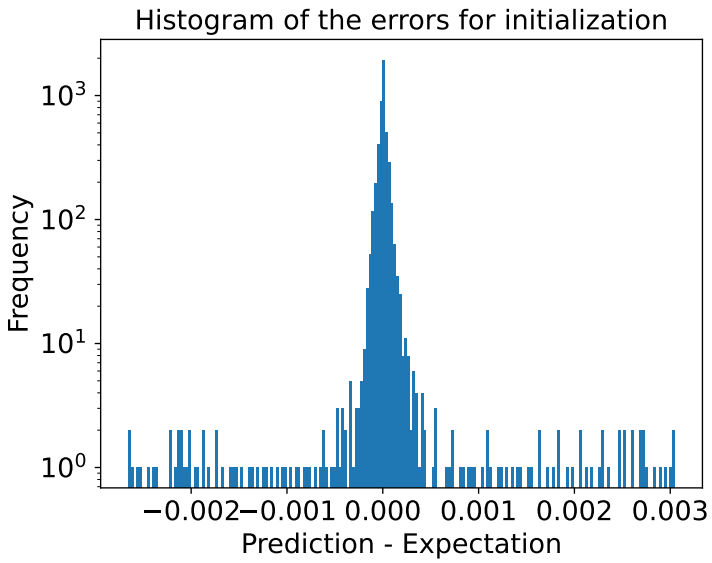


Figure 11: Histogram of the difference between the predictions by the discrete PINN and the solution at the initialization.

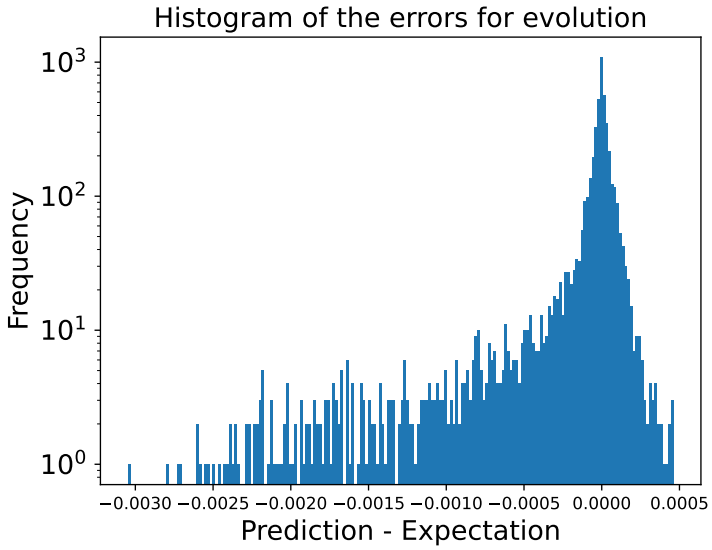


Figure 12: Histogram of the difference between the predictions by the discrete PINN and the solution at the end of the evolution $T = 0.95$.

4.2 Time Continuous Sources

Now that we have demonstrated that the neural network is able to reproduce the simple behavior of a pulse, we can move on to the simulation of time continuous sources in order to evaluate whether the discrete PINN method can reproduce the results of the M1 scheme without its associated artifacts.

We will begin with the test of crossing beams in Section 4.2.1, to verify whether the expected cross pattern is correctly retrieved. Then, in Section 4.2.2, we will reproduce the test with two isotropic sources, in order to confirm that no spurious sources appear between them — a known limitation of the M1 model.

In both tests, the method used to model a time continuous source is the same: at each time step, we add the initial condition again to simulate a source that continuously emits photons.

Finally, in Section 4.2.3, we will investigate the method's sensitivity to hyperparameters, by highlighting the appearance of an artifact in a particular configuration.

4.2.1 Crossing Sources

This section focuses on the test of crossing beams. The system consists of two Gaussian sources with a small angular dispersion. As shown in Figure 13, the first source is located at the bottom of the domain and emits photons upward, while the second source is placed on the left side of the domain and emits photons toward the right. As a result, the two beams cross in the center of the domain. In this case, the equation of the sources is the same as the ones for the gaussians pulses but with an inverse squared spatial dispersion of 70 and a inverse squared velocity dispersion of 30.

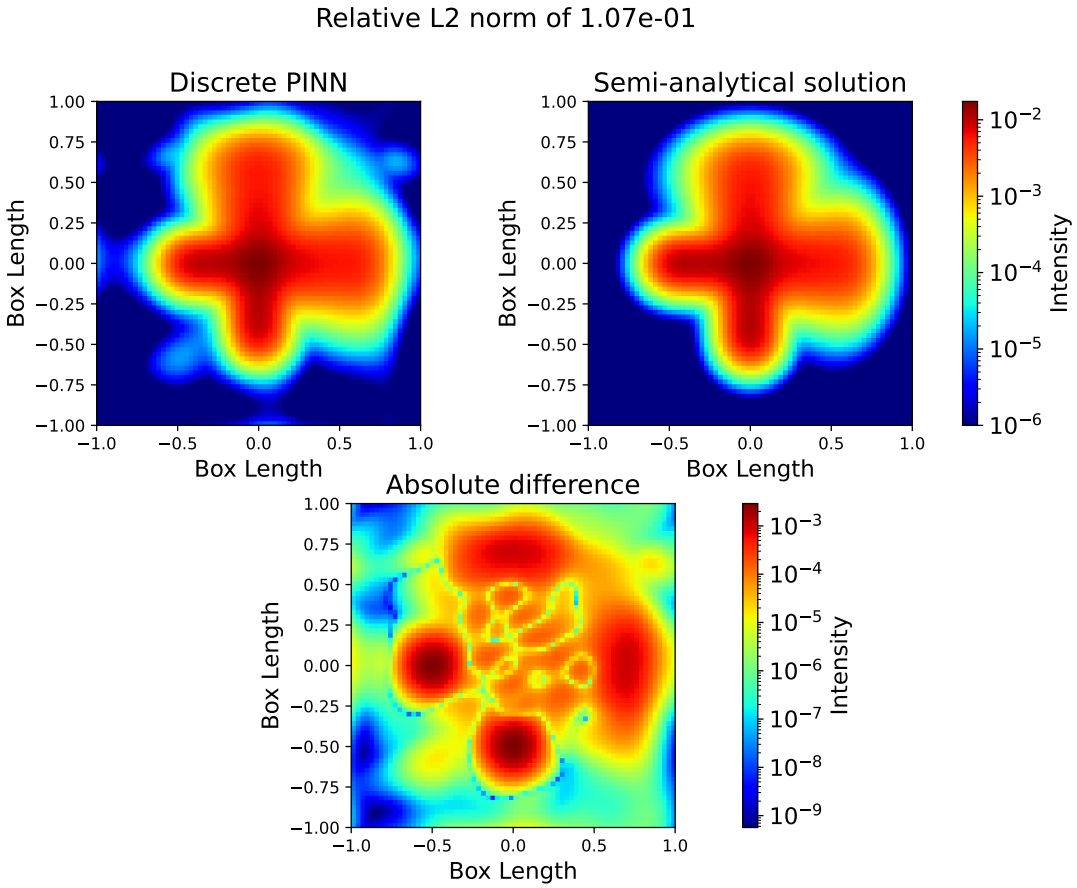


Figure 13: Crossing beams simulated with the discrete PINNs (top left panel) and the semi-analytical solution (top right panel). The bottom panel correspond to the absolute difference of the two images. The L2 norm and the relative L2 norm are showed in the title of the figure.

We present in Figure 13 the results of the two sources described above, following the transport equation (4), as predicted by a discrete PINN. The neural network used in this simulation consists of three layers with 40 neurons each. As explained in Section 3.3, a

sine activation function is used for the hidden layers, and a softplus activation function is applied at the output to ensure positivity, as defined in Equation (17).

The initialization phase consists of 500 iterations with 30,000 randomly sampled collocation points. For the time evolution, we used 100 iterations per time step with 20,000 collocation points and a time step size of 0.05, up to a final time of $T = 1.2$.

The predictions from the discrete PINN are shown in the top left panel of Figure 13. We observe that the crossing of the beams is clearly visible, suggesting that the network qualitatively captures the expected behavior. For comparison, the semi-analytical solution described in Section 3.4 is shown in the top right panel.

The bottom panel displays the absolute difference between the two solutions. We note that the largest discrepancies are on the order of 10^{-3}. The relative L2 norm between the predicted and reference solutions is 1.07×10^{-1}. Although some artifacts appear next to the crossing zone—where the expected values are very small—their magnitude remains low (approximately 4×10^{-5}) compared to the values inside the main beam structure. We can also observe that the largest discrepancies occur near the regions where the photons are initially generated—namely at the left and bottom boundaries of the domain—and at the terminal regions where the photons reach after propagating along their respective beams, i.e., at the top and right boundaries.

The relative L2 norm is computed as follows:

$$L_2 = \sqrt{\frac{\sum_{j=1}^{N_{points}} (I_\theta - I_{analytical})^2}{\sum_{j=1}^{N_{points}} I_{analytical}^2}} \tag{23}$$

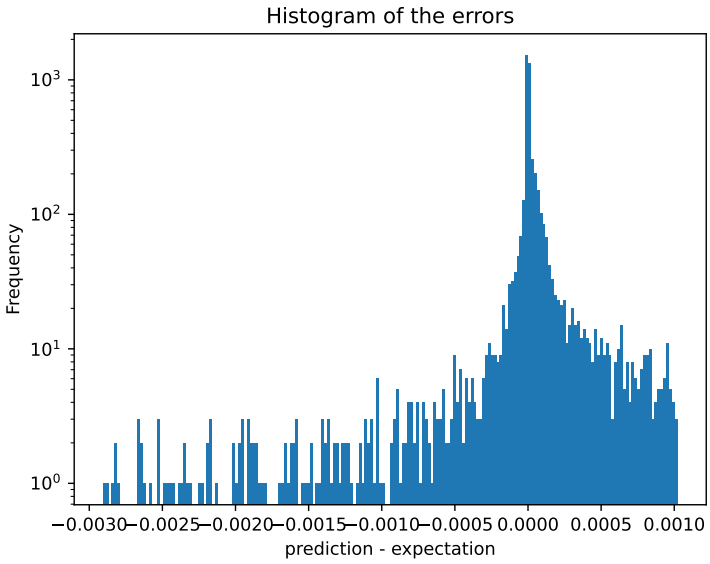


Figure 14: Histogram of the differences between the predictions of the neural network and the semi-analytical solution for the crossing sources.

The histogram shown in figure 14 illustrates the distribution of the prediction errors. We can observe a tail of negative errors and a clump of errors at the right of the main peak. However, the order of magnitude of the differences remains small, with most errors lying between -1×10^{-3} and 1×10^{-3}.

Figure 15 presents the evolution of the relative L^2 error at each time step. The error increases slightly at the beginning of the simulation and eventually converges to a value just below 1.25×10^{-1}. Despite this increase, the L^2 norm remains on the order of $1\text{-}3 \times 10^{-1}$ throughout the simulation, indicating that the discrete PINN remains stable across all time steps.

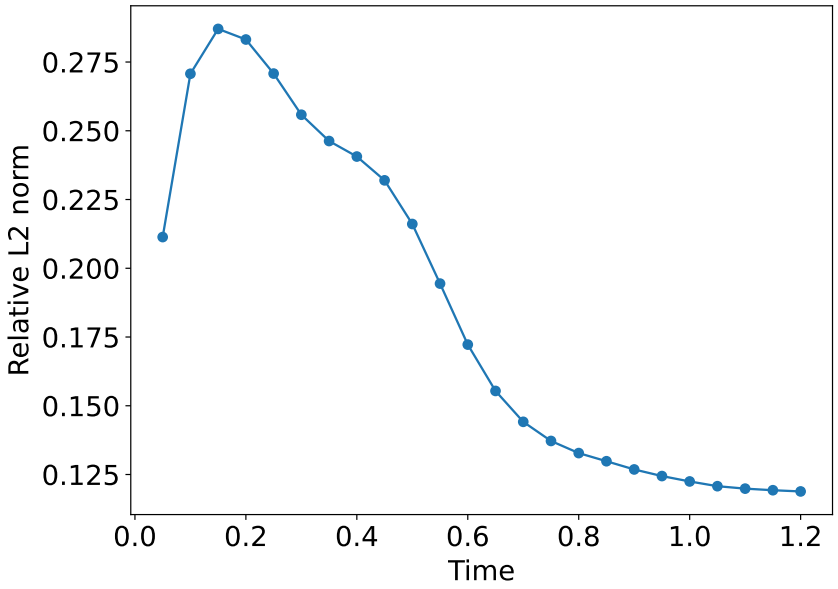


Figure 15: Evolution of the relative L2 norm relative to the time step of the simulation.

4.2.2 Isotropic Sources

The second test involves two isotropic sources, and the objective is to determine whether pseudo-sources appear, as observed in the left panel of figure 4 representing the solution given by the M1 method. The results of this simulation are presented in figure 16. In this case, we use the same neural network architecture and parameters as in the previous test. However, the final simulation time is set to 0.5 to ensure that the two photon fronts have had time to reach and interact with each other and the time step is set to 0.02.

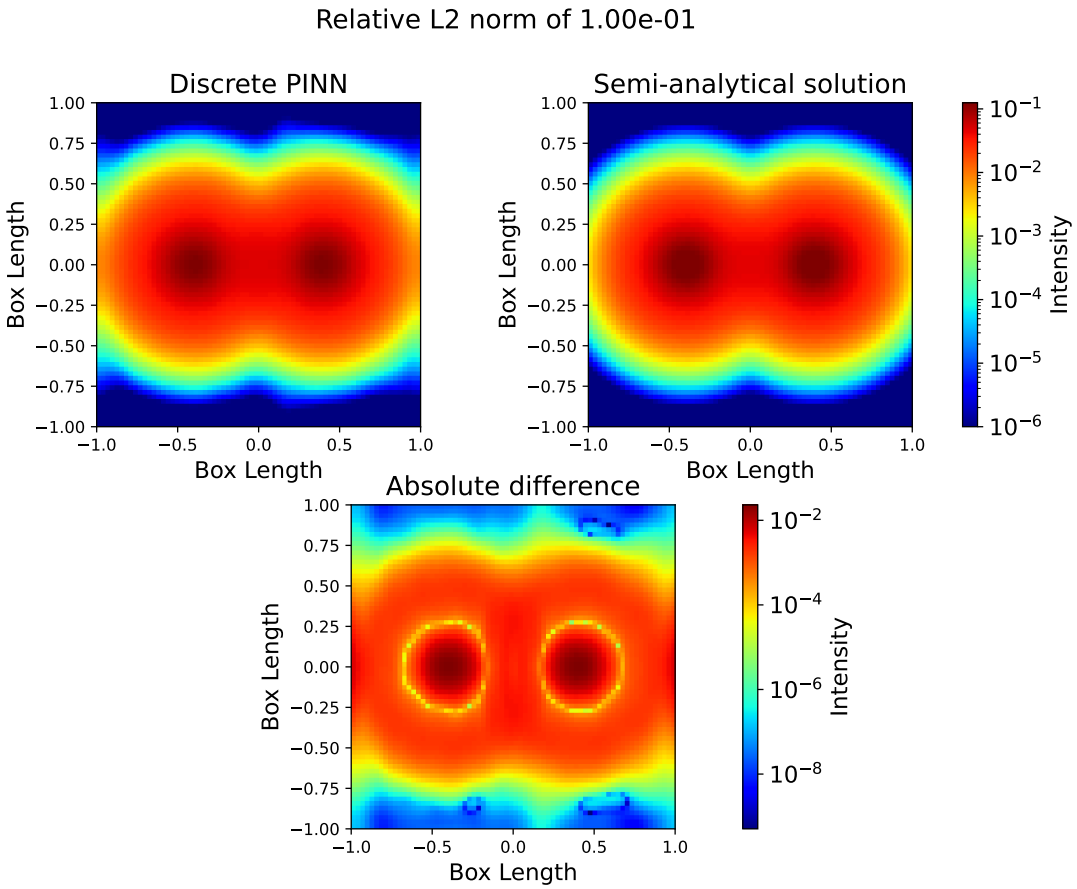


Figure 16: Isotropic Sources simulated with the discrete PINNs (top left panel) and the semi-analytical solution (top right panel). The bottom panel correspond to the absolute difference of the two images. The L2 norm and the relative L2 norm are showed in the title of the figure.

As in the previous test, we observe that the prediction provided by the discrete PINN agrees well with the semi-analytical solution, both in terms of the general shape and the order of magnitude of the results. The relative L2 norm for this test is 1.0×10^{-1}, which is consistent with the value obtained in the crossing beam case.

As before, we provide a map of the absolute difference between the neural network prediction and the expected solution. We can see that the order of magnitude of this difference is around 10^{-2} near the centers of the sources, and decreases to approximately 10^{-3} in the region between the source centers and the photon fronts.

Moreover, we can observe concentric circular patterns around the two source centers. These structures suggest that the PINN tends to produce smoother gradients within the source regions, compared to the sharper profiles obtained with the semi-analytical solution.

In Figure 17, we present the histogram of errors for this test. As in the previous cases, we observe that the predictions are generally underestimated compared to the expected values. However, the errors remain within the same order of magnitude, mostly ranging between -5×10^{-3} and 5×10^{-3}. This confirms that, despite the slight bias toward underestimation and a second peak at the right of the main peak, the neural network remains quantitatively consistent across the entire domain. The interpretation of the mentionned secondary peak is still unclear.

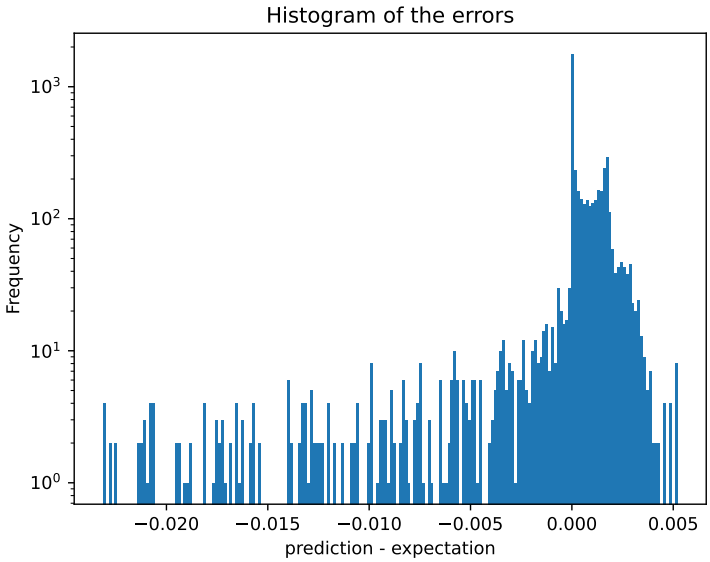


Figure 17: Histogram of the differences between the predictions of the neural network and the semi-analytical solution for the isotropic sources.

As in the previous test, we observe an increase in the relative L2 norm, as shown in Figure 18. However, since the total simulation time is shorter than in the case of the crossing sources, we do not observe the same convergence behavior. In this case, the relative L2 norm increase until the convergence to a value of approximately 1×10^{-1}, and the final value is comparable to that obtained in the crossing beams test. This allows us to conclude that the discrete PINN remains stable throughout the entire simulation.

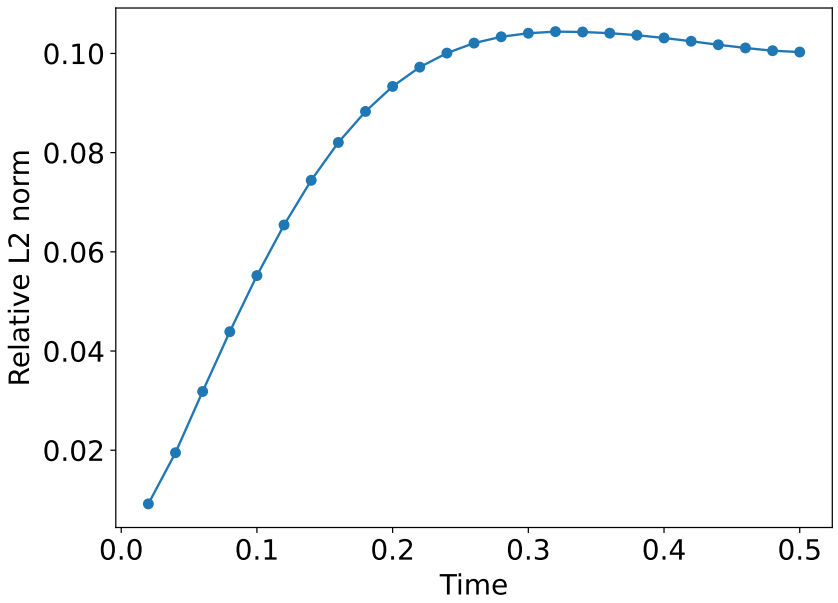


Figure 18: Evolution of the relative L2 norm relative to the time step of the simulation.

4.2.3 Sensitivity to the parameters

Time step size During the optimization of the hyperparameters, we encountered certain configurations that led to unexpected results. For instance, in the case of the crossing

beams described in Section 4.2.1, we initially identified a set of hyperparameters that minimized the final relative L2 error. In an attempt to further improve the performance, we tested the effect of reducing the time step, expecting a more accurate solution due to the finer temporal resolution.

However, instead of the expected improvement, the result obtained is shown in Figure 19, and it clearly deviates from the expected physical behavior.

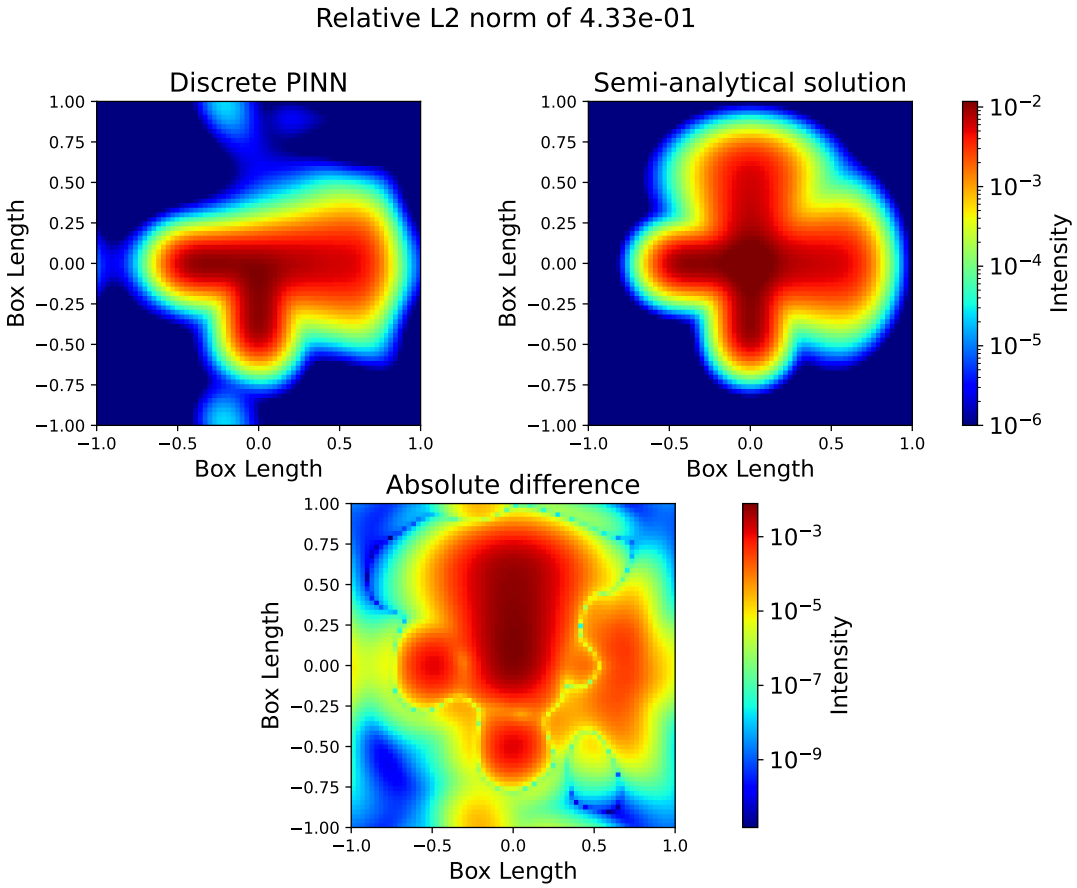


Figure 19: Evolution of the cross beams with a time step equal to 0.02.

We can observe in Figure 19 that the beam originating from the bottom of the domain is unexpectedly stopped during the evolution and does not reach the top of the box, contrary to what is expected. This behavior is not the result of the typical artifacts observed with the M1 method, but rather an unexpected limitation of the neural network in this specific configuration. The cause of this behavior remains unclear. Interestingly, this issue appears only in the crossing beams test case, and was not observed in the other configurations tested.

Figure 20 shows the evolution of the relative L2 norm over time. We can see that the error is significantly higher from the beginning of the simulation, compared to the standard case discussed in Section 4.2.1, indicating that the discrepancy arises early during the evolution and persists throughout.

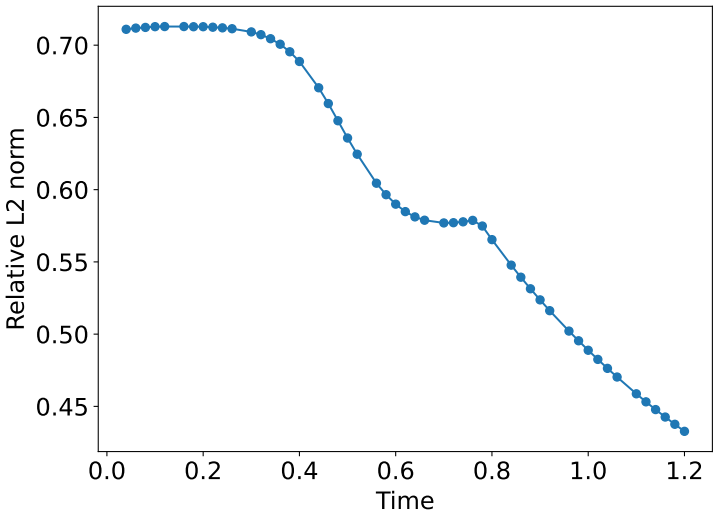


Figure 20: Evolution of the relative L2 norm for the case of the crossing beams with a time step of 0.02.

Neural Network size Still in the context of the crossing beams, we also tested a simulation using fewer neurons. In this test, the network is composed of three layers with twenty neurons each, and only twenty iterations are used for the evolution optimization step. The simulation runs in approximately ten minutes and yields a final relative L2 norm of 2.13×10^{-1}. However, when analyzing the error distribution, we observed an unexpected secondary peak. This histogram is shown in Figure 21. We can see a noticeable peak around -0.002, the origin of which remains unclear.

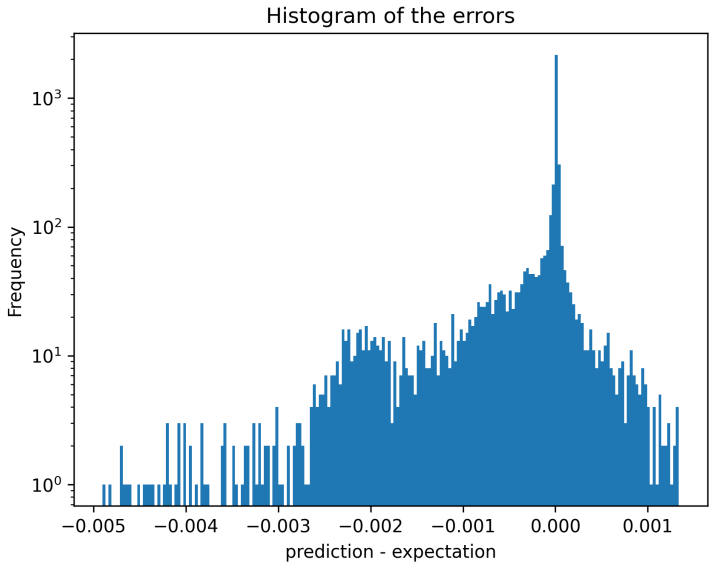


Figure 21: Histogram of the differences between the predictions of the neural network and the semi-analytical solution for the crossing sources with a neural network composed of three layer of twenty neurons.

5 Discussion

We have tested the discrete PINN method on simple cases where the M1 scheme—one of the most widely used methods for solving the radiative transfer equation—exhibits well-known artifacts. The first test case, involving crossing beams, highlights how M1 tends to merge the two beams into a single flux, representing the sum of the two initial fluxes. In the second case, involving two isotropic sources, M1 produces a spurious pseudo-source at the location where the two I-fronts collide. In both scenarios, the discrete PINN method produces qualitatively correct results and avoids the artifacts observed with M1, while maintaining reasonable accuracy.

Additionally, in Section 4.1, we evaluated the method's ability to reproduce the behavior of time-dependent Gaussian pulses. The results demonstrated that the discrete PINN approach achieves stable and consistent accuracy throughout the simulation.

It is important to note that the computational time for the simulations done in Section 4.2 can be important. For instance, the isotropic source simulation takes approximately 15 minutes, while the crossing beams case requires around 30 minutes on a GPU NVIDIA H100 NVL from the HPC. This large runtime mainly results from the choice of a relatively large neural network architecture, composed of three layers with forty neurons each, which increases the number of parameters to optimize. In addition, the number of iterations during each evolution step was set to 100 in our configuration, further contributing to the total computation time.

To investigate this, we tested a lighter configuration using three layers of 20 neurons while keeping 100 iterations per evolution step. In this case, the runtime dropped significantly—around 5 minutes for the isotropic sources and 10 minutes for the crossing beams—while the final relative L2 error remained around 2×10^{-1}. These results suggest that further optimization of the architecture could lead to better compromises between computational cost and accuracy.

A quantitative study of the memory footprint of the method would also be valuable in future work, especially in comparison with traditional radiative transfer solvers.

In future developments, the discrete PINN method could be hybridized with classical methods such as M1. For example, the radiative transfer equation solution can be analytically written as the M1 solution plus a correction g. We can then compute the M1 solution using classical numerical explicit methods such as Aubert & Teyssier (2008) while the correction g could be obtained via a discrete PINN as we do here. Eventually, we can obtain accurate results within a reasonable computational time, unlike higher-order methods such as Pn. It would also be interesting to benchmark the performance of this hybrid approach using standard radiative transfer test problems, such as those proposed for reionization simulations in Shapiro et al. (2012), but this requires coupling our radiative transfer solver with a chemistry solver accounting for the interaction between the ionizing photons and the HI gas.

6 Conclusion

To conclude, the discrete PINN method provides qualitatively accurate results while successfully avoiding the artifacts typically introduced by low order schemes such as M1. Although the computational time remains a limiting factor, it could be mitigated through better tuning of the network architecture. It is also worth noting that, as highlighted in the introduction, neural networks are inherently well-suited for higher-dimensional prob-

lems, suggesting that the computational cost may not scale prohibitively when extending the method to three spatial dimensions.

Looking ahead, a promising direction would be to combine the discrete PINN with the M1 method, using the latter to provide a fast but approximate solution, and the former to correct its deficiencies. This hybrid approach could improve both accuracy and efficiency, making discrete PINNs a valuable tool for future radiative transfer simulations. Recent exascale astrophysical numerical simulation codes such as Dyablo would be a relevant target for implementation of this method.

A Slurm job script

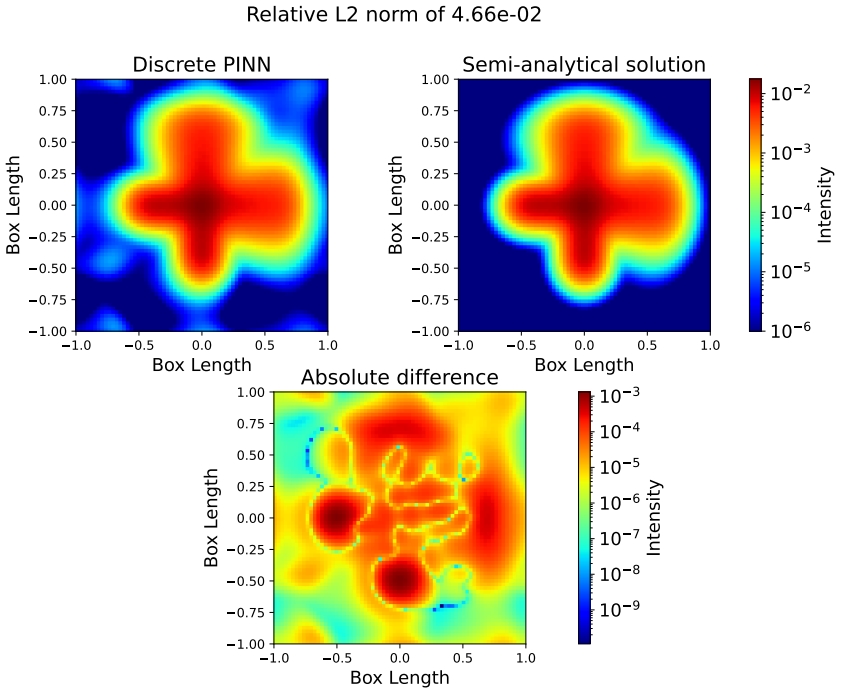
Here is the script in the file `launch_RT_croise.txt`:

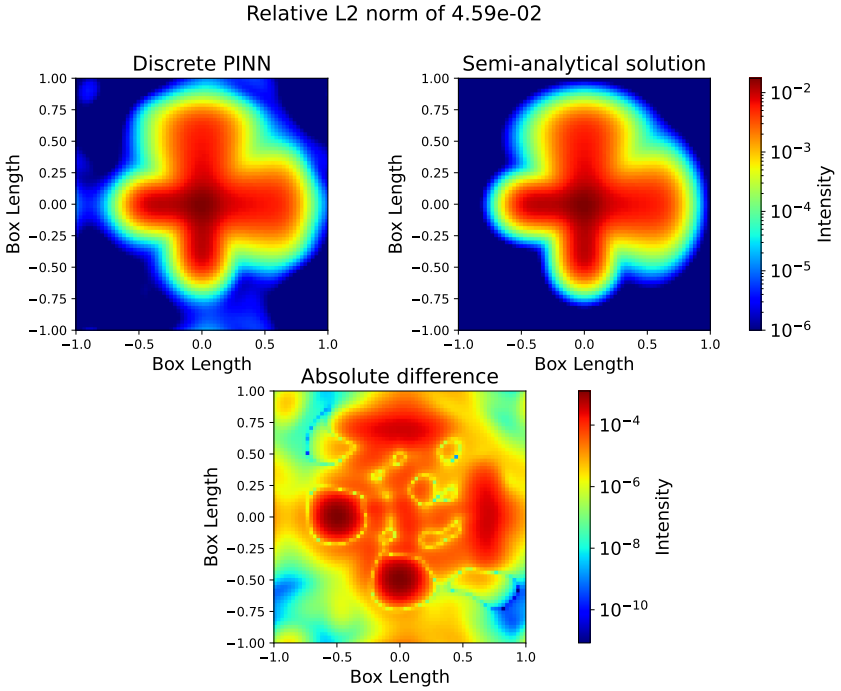
```
#!/bin/sh

#SBATCH -p publicgpu
#SBATCH -N 1
#SBATCH --gres=gpu:4
#SBATCH --exclusive
#SBATCH -t 01:00:00
#SBATCH --constraint=gpudp
#SBATCH --constraint=gputc
#SBATCH --mail-type=END
#SBATCH --mail-user=maxime.gressier@etu.unistra.fr

python ./scimba/examples/time_discrete/radiativetransfert_sources_croisees.
    py
```

B Additional plots from the end of the internship

B.1 Tests for the Crossed Sources with 0.02 time step

Solution: Remove the predefined seed.


Relative L2 norm of 4.66e-02
Discrete PINN
Semi-analytical solution
Absolute difference
Box Length
Intensity


Relative L2 norm of 4.59e-02
Discrete PINN
Semi-analytical solution
Absolute difference
Box Length
Intensity

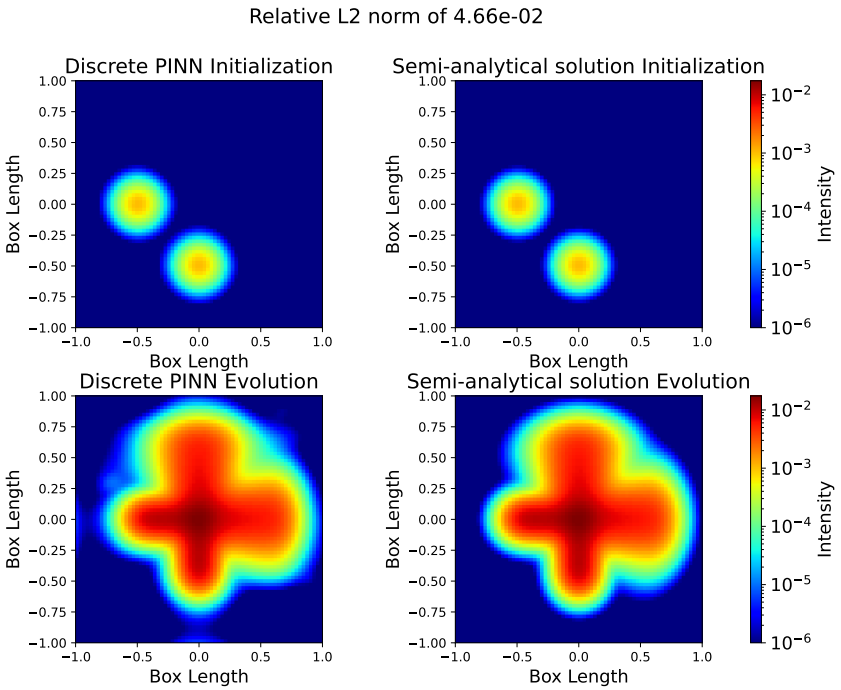
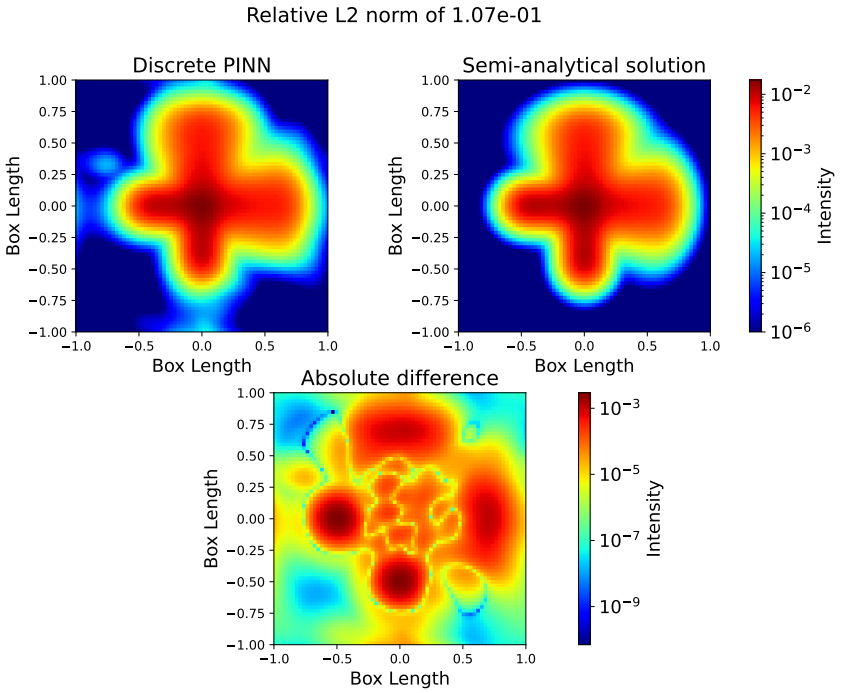


Figure 22: Three layers of 40 neurons and 0.02 time step.



B.2 Tests on the size (spatial and angle dispersion) of the source

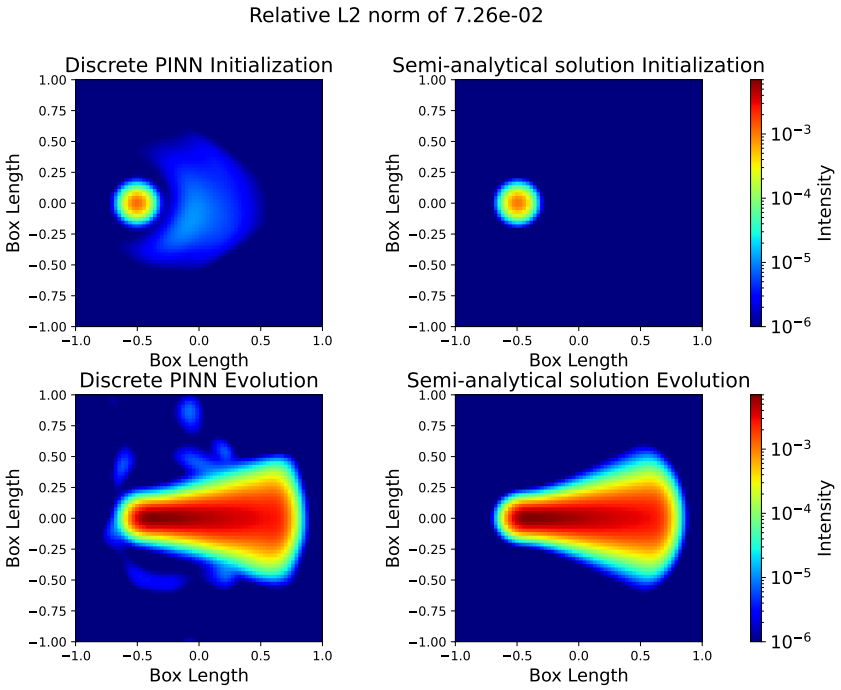


Figure 23: spatial dispersion $(1/\sigma^2) = 150$; angle dispersion $(1/\sigma^2) = 30$

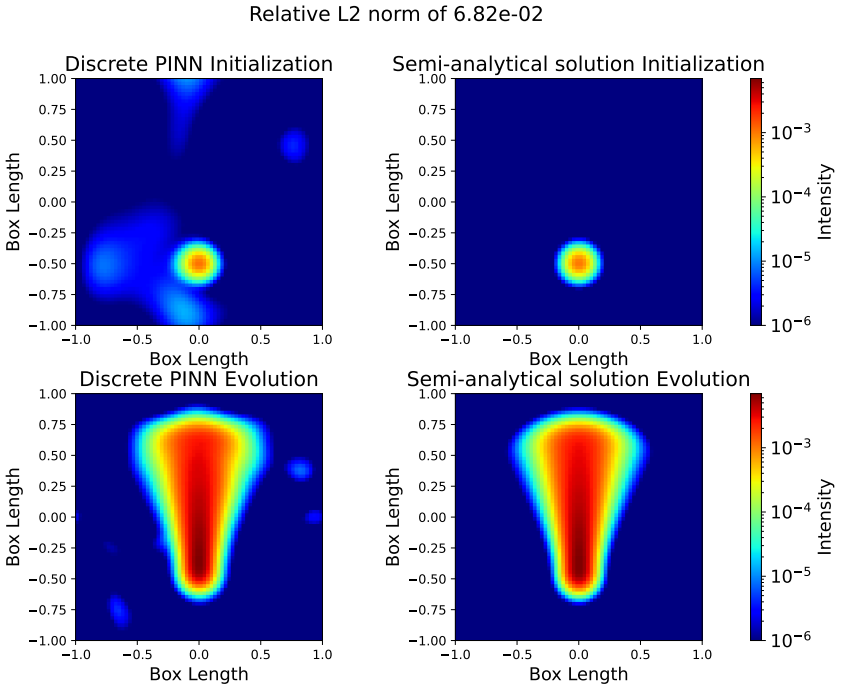


Figure 24: spat disp = 150; angle disp = 30

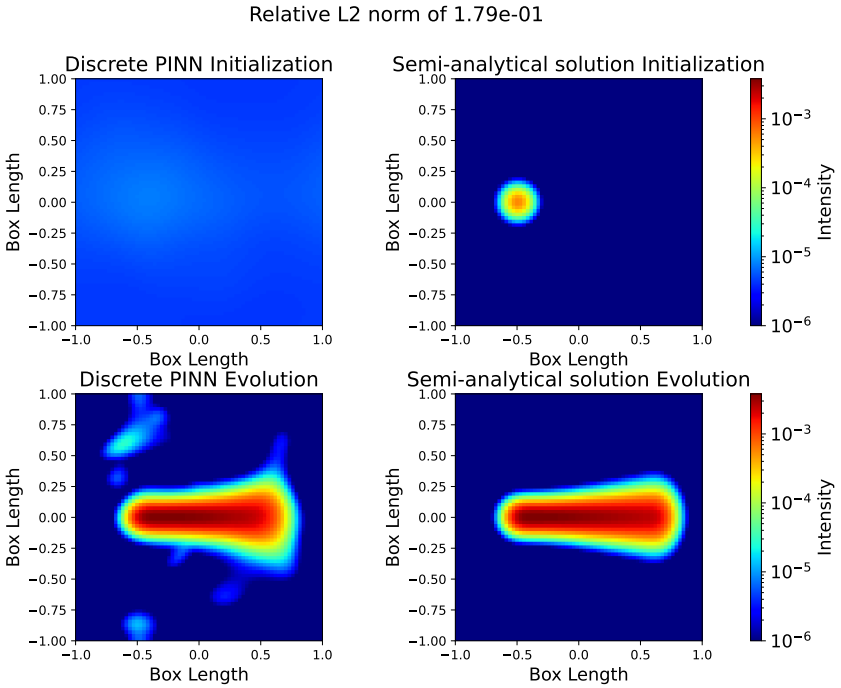


Figure 25: spat disp = 150; angle disp = 100

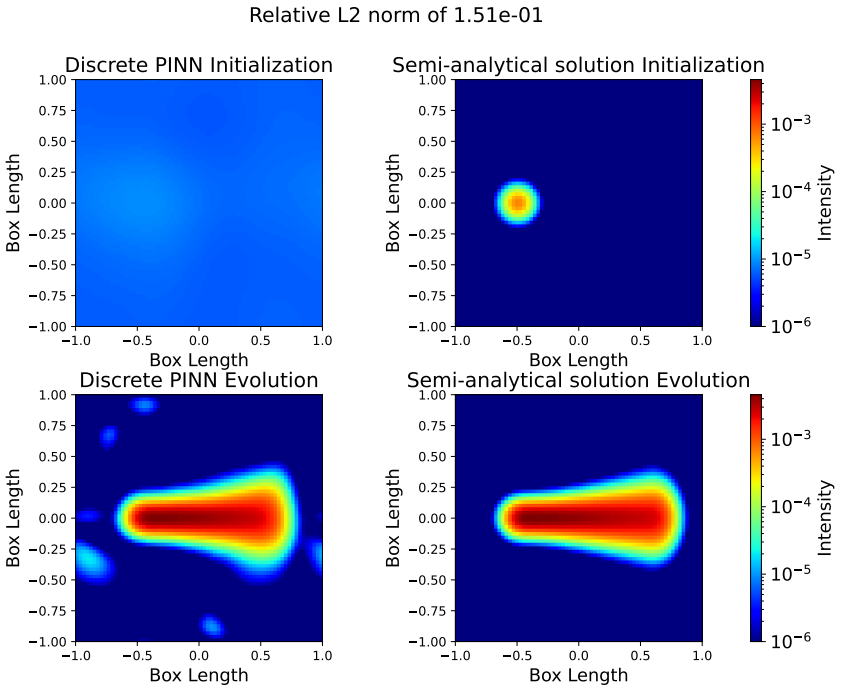


Figure 26: spat disp = 150; angle disp = 70

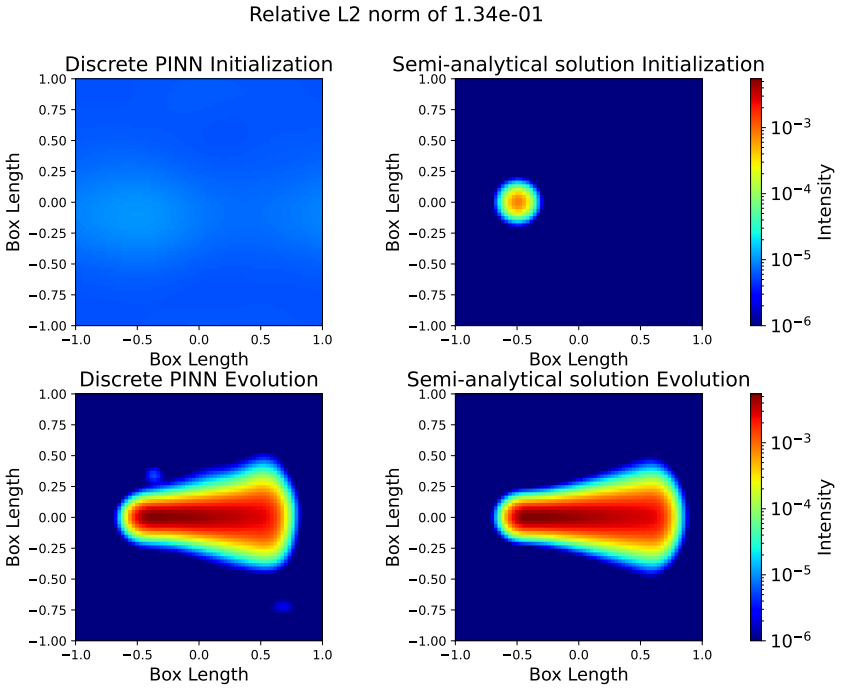


Figure 27: spat disp = 150; angle disp = 50

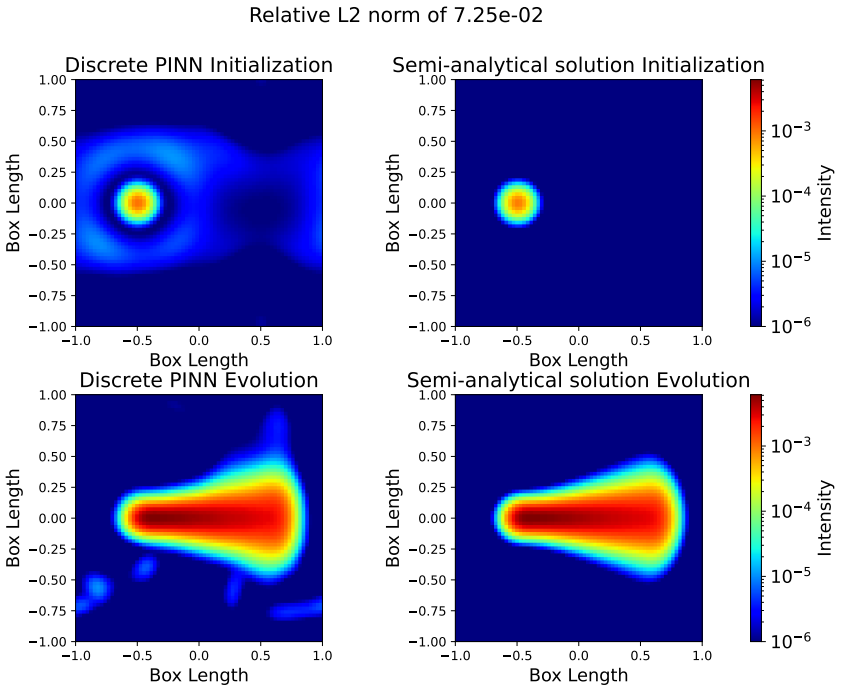


Figure 28: spat disp = 150; angle disp = 40

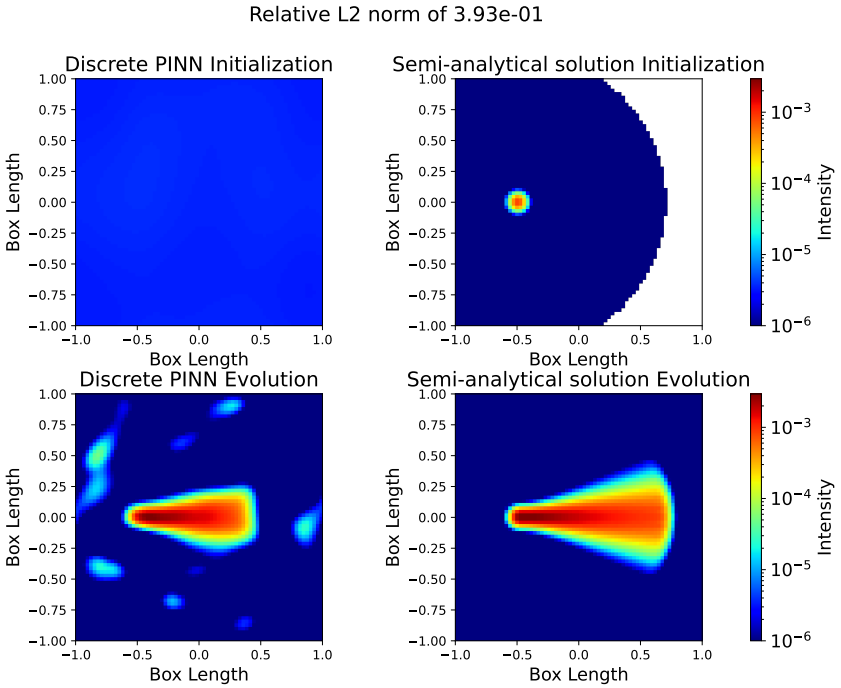


Figure 29: spat disp = 500; angle disp = 30

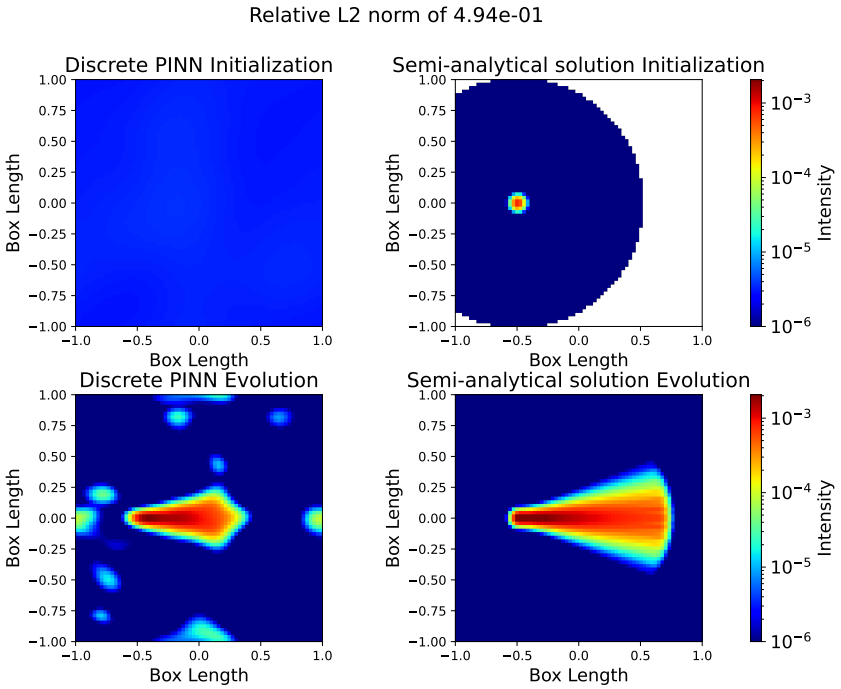


Figure 30: spat disp = 700; angle disp = 40

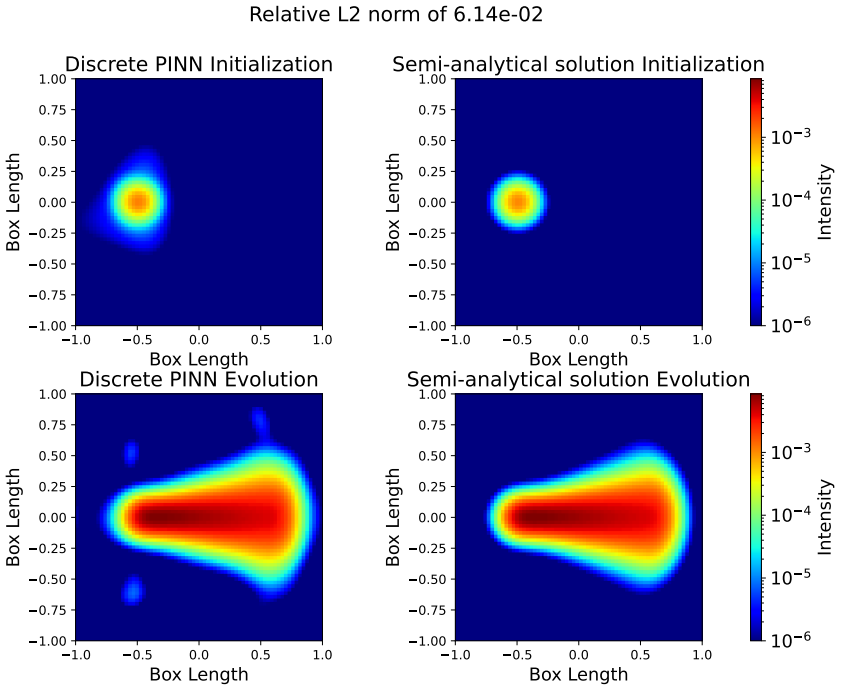


Figure 31: spat disp = 100; angle disp = 30

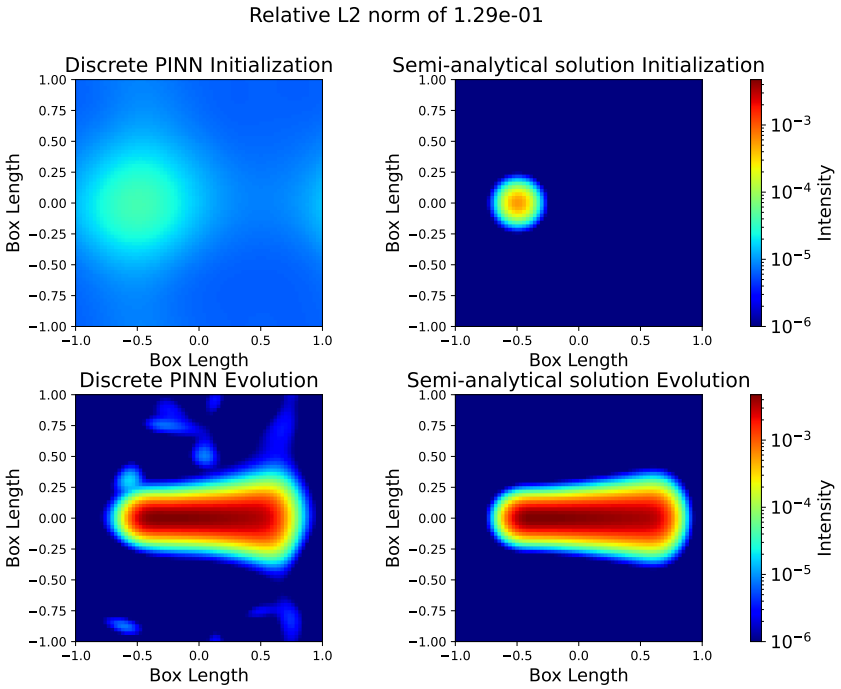


Figure 32: spat disp = 100; angle disp = 100

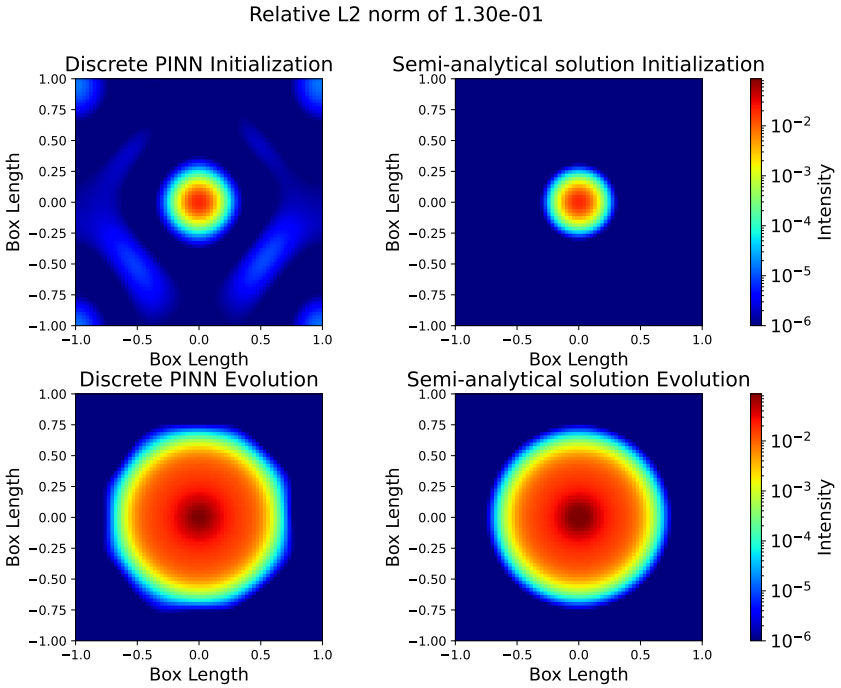


Figure 33: spat disp = 100

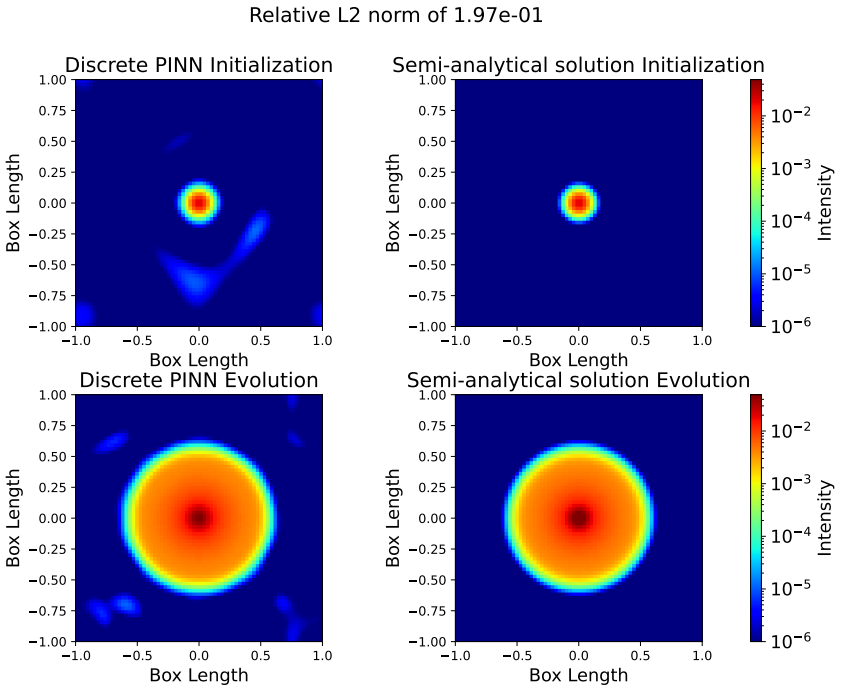


Figure 34: spat disp = 200

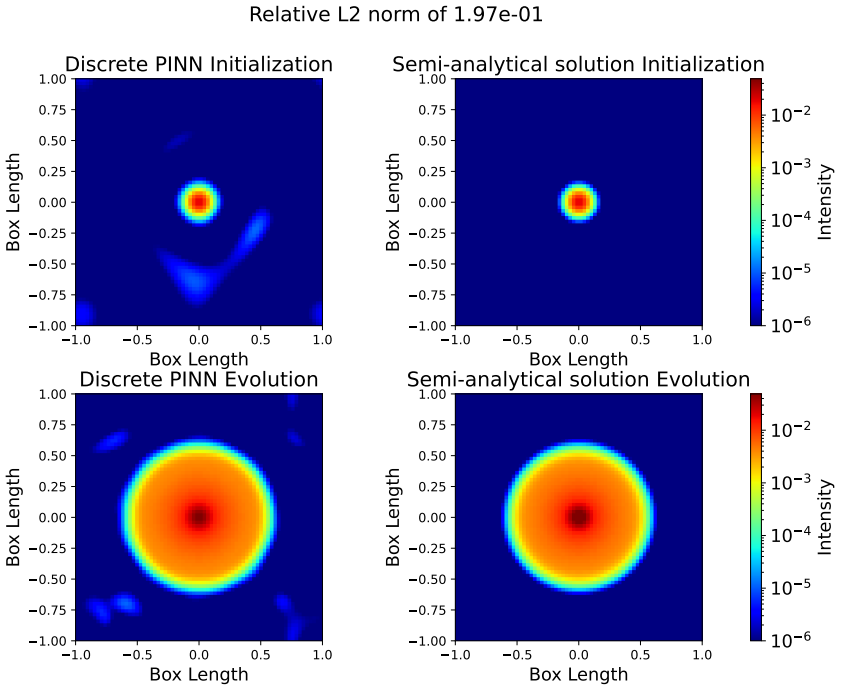


Figure 35: spat disp = 300

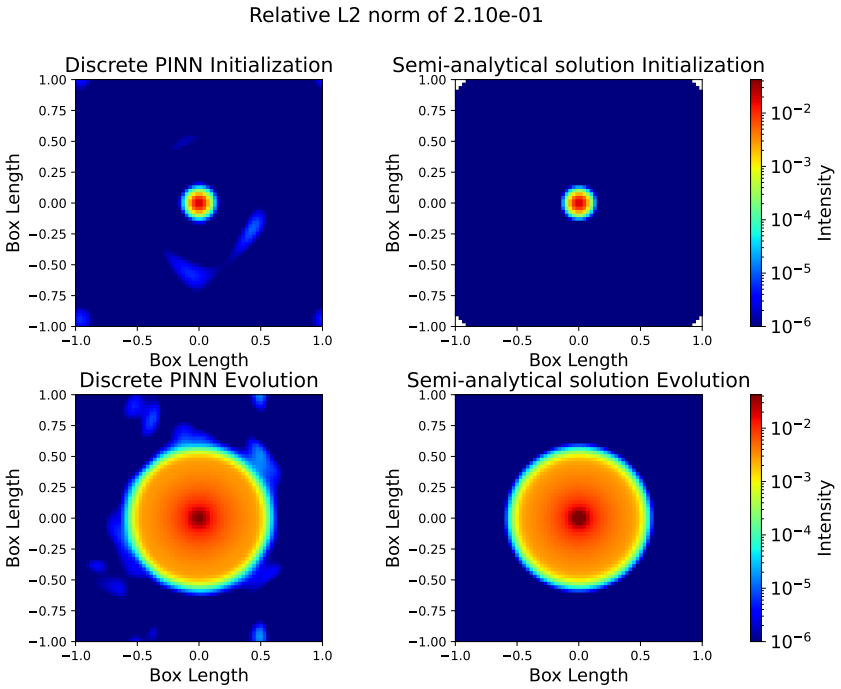


Figure 36: spat disp = 400

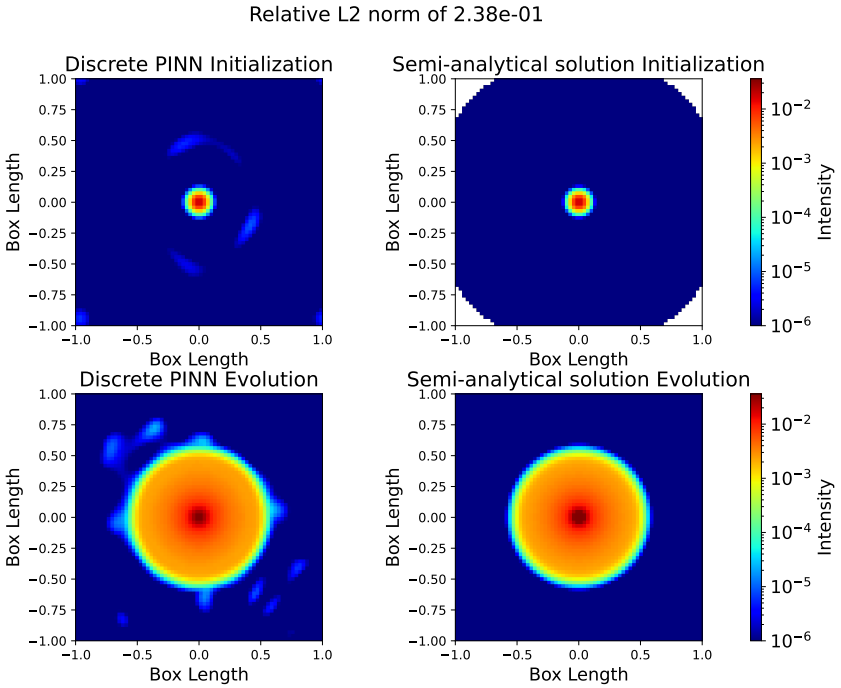


Figure 37: spat disp = 500

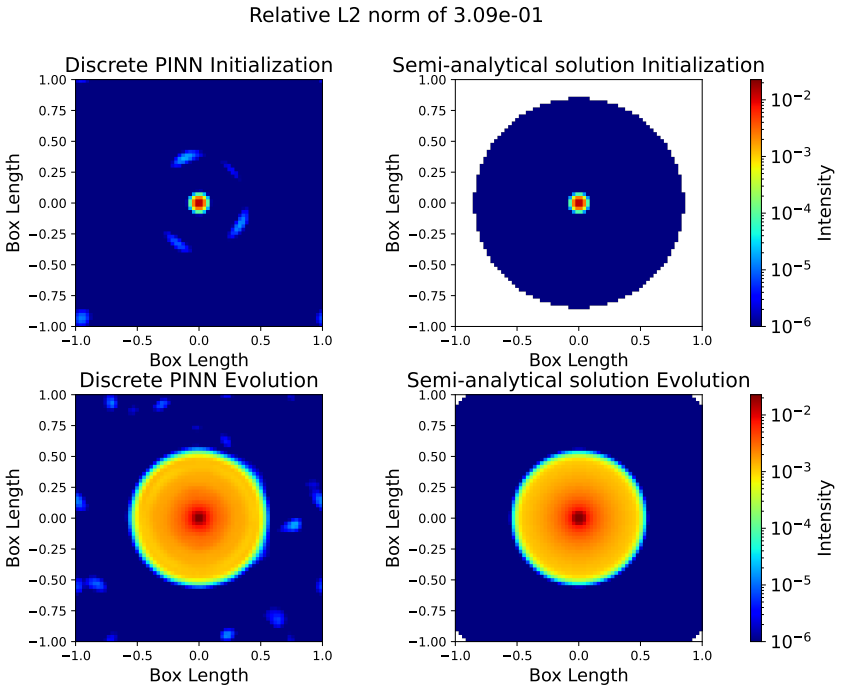


Figure 38: spat disp = 1000

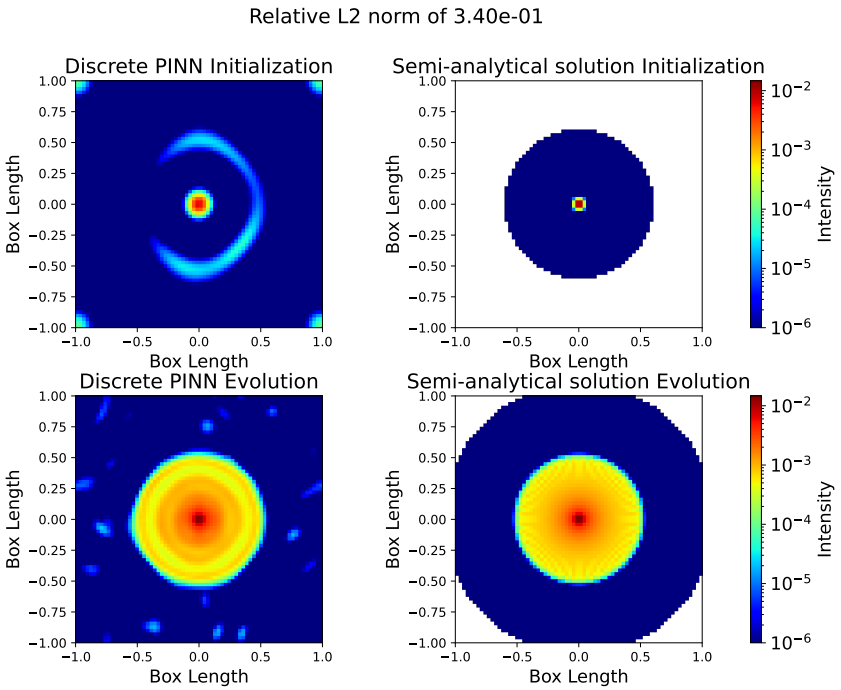
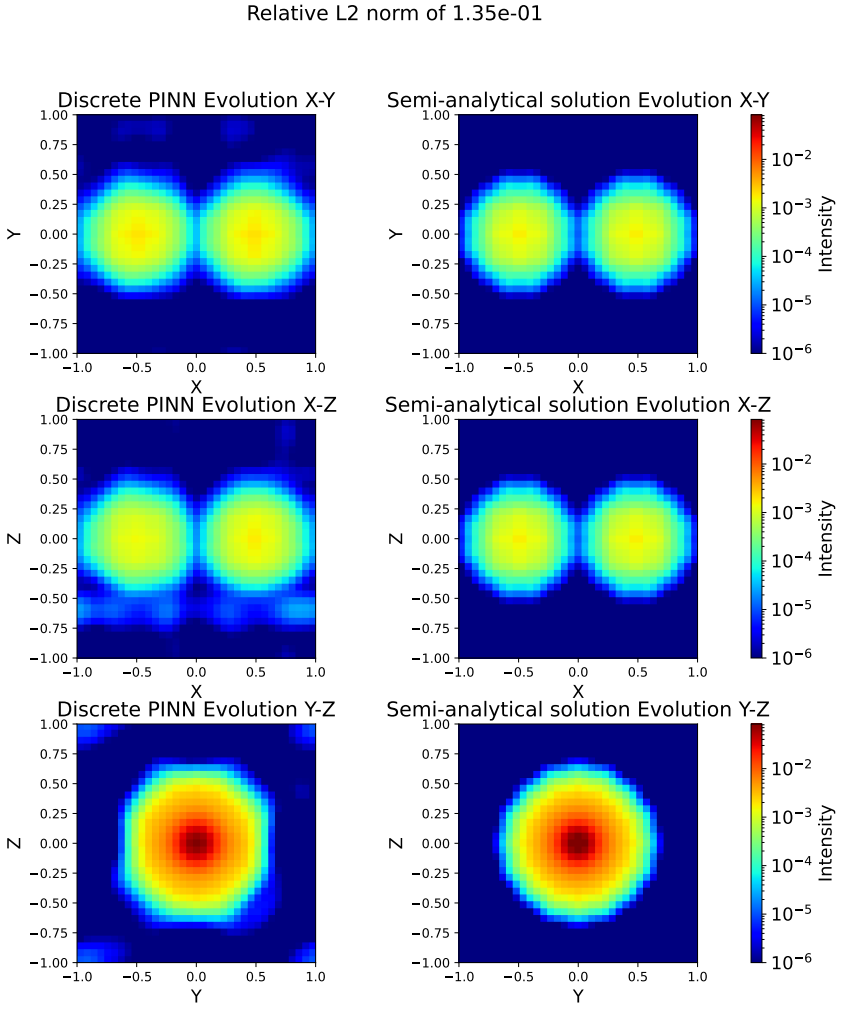
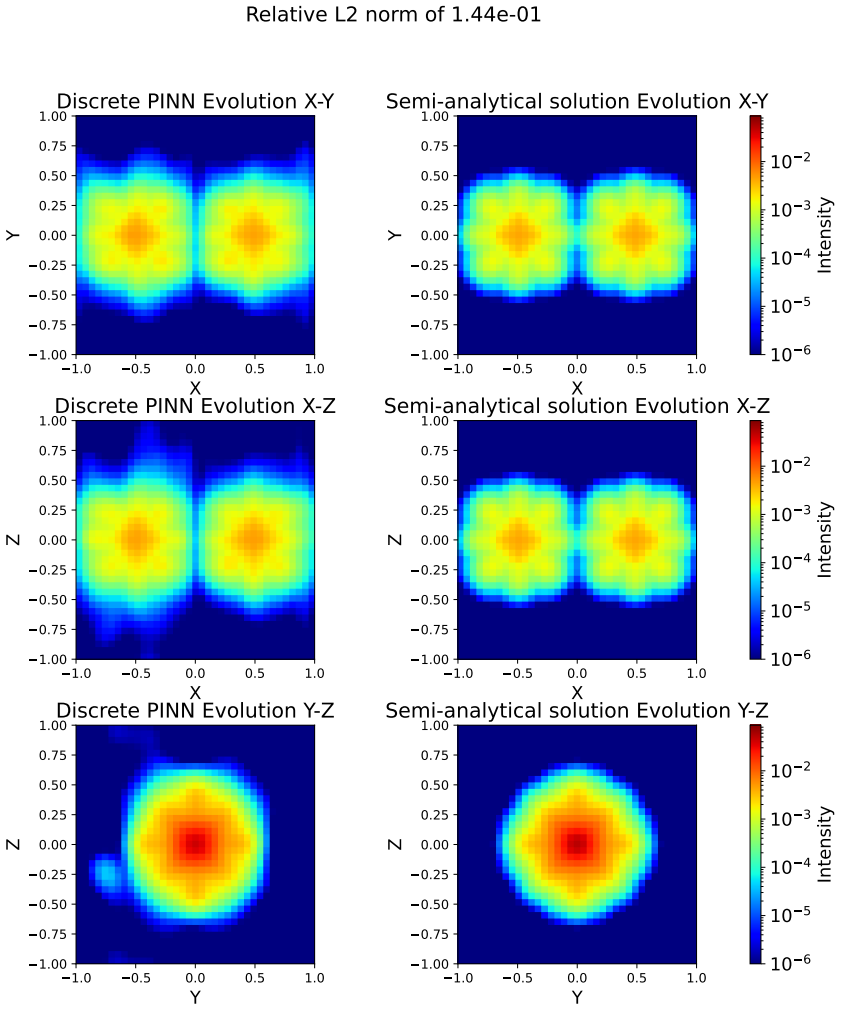


Figure 39: spat disp = 2000

B.3 Test in 3D for isotropic sources





References

Adams, N. J., Conselice, C. J., Ferreira, L., et al. 2022, Monthly Notices of the Royal Astronomical Society, 518, 4755–4766

Aghanim, N., Akrami, Y., Ashdown, M., et al. 2020, Astronomy & Astrophysics, 641, A6

Allaire, G., Blanc, X., Després, B., & Golse, F. 2018, Transport et diffusion, ed. E. de l'Ecole polytechnique, 328

Atek, H., Shuntov, M., Furtak, L. J., et al. 2022, Monthly Notices of the Royal Astronomical Society, 519, 1201

Aubert, D. & Teyssier, R. 2008, Monthly Notices of the Royal Astronomical Society, 387, 295–307

Barkana, R. & Loeb, A. 2001, Physics Reports, 349, 125–238

Becker, R. H., Fan, X., White, R. L., et al. 2001, The Astronomical Journal, 122, 2850–2857

Beltran-Pulido, A., Bilionis, I., & Aliprantis, D. 2022, IEEE Transactions on Energy Conversion, 37, 2678–2689

Bosman, S. E. I., Davies, F. B., Becker, G. D., et al. 2022, Monthly Notices of the Royal Astronomical Society, 514, 55–76

Chen, S., Shan, B., & Li, Y. 2023, Efficient Discrete Physics-informed Neural Networks for Addressing Evolutionary Partial Differential Equations

Franck, E., Michel-Dansac, V., & Navoret, L. 2025, Neural semi-Lagrangian method for high-dimensional advection-diffusion problems

Helton, J. M., Rieke, G. H., Alberts, S., et al. 2025, Nature Astronomy, 9, 729–740

Lamb, D. Q. & Haiman, Z. 2003, Gamma-Ray Bursts as a Probe of the Epoch of Reionization

Levermore, C. 1984, Journal of Quantitative Spectroscopy and Radiative Transfer, 31, 149

Nurbekyan, L., Lei, W., & Yang, Y. 2023, Efficient Natural Gradient Descent Methods for Large-Scale PDE-Based Optimization Problems

Ocvirk, P., Aubert, D., Sorce, J. G., et al. 2020, Monthly Notices of the Royal Astronomical Society, 496, 4087–4107

Palanque, M., Ocvirk, P., Frank, E., & Gerhard, P. 2025, in prep

Raissi, M., Perdikaris, P., & Karniadakis, G. 2019, Journal of Computational Physics, 378, 686

Shapiro, P. R., Iliev, I. T., Mellema, G., et al. 2012, in AIP Conference Proceedings (AIP), 248–260

Shimabukuro, H., Hasegawa, K., Kuchinomachi, A., Yajima, H., & Yoshiura, S. 2022, Publications of the Astronomical Society of Japan, 75, S1–S32

Zhang, B., Cai, G., Weng, H., et al. 2023, Machine Learning: Science and Technology, 4, 045015